Universidad Carlos III de Madrid
Escuela Politécnica Superior

ESTUDIO DE LA IMPORTANCIA DEL MODELADO EN PLANIFICACIÓN AUTOMÁTICA

Proyecto Fin de Carrera
Ingeniería Técnica en Informática de Gestión

Autor: Ángela Barroso Peña
Tutor: Ángel García Olaya

Título: Estudio de la importancia del modelado en la planificación automática
Autor: Ángela Barroso Peña

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mi tutor, por darme esta oportunidad y por su gran ayuda.

A mis padres, por su infinita paciencia y apoyo siempre incondicional.
A mi hermano, grandísimo apoyo.

A Rubén, que no ha dejado de ayudarme y darme ánimos.
Sin su apoyo esto no hubiera sido posible.

A mi grupo de Informatilocos y amigos, por todo lo que hemos compartido en la universidad y seguimos compartiendo fuera de ella.

Gracias a todos.

Resumen

La temática de investigación del presente proyecto está relacionada con el área informática de Inteligencia Artificial, y más concretamente de la Planificación Automática. Esta última, trata de encontrar una secuencia de acciones que, partiendo de un estado inicial, permita llegar a un estado final u objetivo.

La finalidad de este trabajo es estudiar la importancia del modelado en la Planificación Automática, mediante modificaciones del dominio de planificación especificadas en un lenguaje de definición denominado PDDL (Planning Domain Description Language, o Lenguaje de Definición de Dominios de Planificación), que se ha convertido en el lenguaje estándar para describir los problemas de planificación.

Este lenguaje, basado originalmente en STRIPS y ADL pero sensiblemente más expresivo, permite modelar acciones simples como listas de precondiciones y efectos, entendiendo dichas listas como una conjunción de proposiciones. Además, provee características adicionales que definen varios niveles de expresividad y permiten enriquecer la definición de dominios y problemas de planificación.

De este modo, incrementando las especificaciones del dominio, es esperable que los planificadores se vean afectados de manera que la resolución de los problemas de planificación tenga menor coste computacional (de tiempo y recursos).

La experimentación sobre el modelado presentada se realiza sobre distintas instancias o versiones del dominio 'Elevators', dominio de la Competición Internacional de Planificación de 2011. Cada instancia del problema es modelada en PDDL empleando los distintos requerimientos y características del lenguaje.

Abstract

The researched topic in this project is related to the computer area of Artificial Intelligence, and more specifically to Automated Planning. The latter, tries to find a sequence of actions that, starting from an initial state, will lead to a final state or goal.

The purpose of this work is to study the importance of modeling in Automatic Planning, through modifications specified in a planning domain definition language called PDDL (Planning Domain Description Language), which has become the standard language to describe planning problems.

This language, originally based on STRIPS and ADL but significantly more expressive, allows modeling simple actions using a preconditions and effects list, understanding these lists as a conjunction of propositions. It also provides additional features defining expressiveness levels and allowing to enrich the definition of domains and problems of planning.

Thus, with more expressive domain specifications, planners are expected to solve the problems of planning with lower computational cost (time and resources).

The experimentation on the modeling has been done on different instances or versions of the 'Elevators' domain, from the International Planning Competition 2011. Each instance of the problem is modeled in PDDL by using different requirements and features of the language.

Índice general

Índice de figuras

Índice de tablas

1. Introducción y objetivos

Este apartado del documento muestra un resumen sobre el ámbito en el que se ha desarrollado este Proyecto Fin de Carrera, su objetivo y la estructuración de la memoria.

1.1 Visión general

El presente proyecto se enmarca en el área de Inteligencia Artificial, más concretamente en la Planificación Automática. A lo largo de este trabajo obtendremos un acercamiento a la Planificación Automática, al lenguaje de definición de dominios y problemas de planificación, PPDL, y sus diferentes versiones, así como los planificadores que interpretan este lenguaje y compiten para optimizar la resolución de dichos problemas de planificación.

En primer lugar, se realizará un análisis de los dominios y un estudio comparativo de los planificadores más destacados según la categoría de la Competición Internacional de Planificación en que compiten para, posteriormente, pasar al modelado de los dominios de planificación para medir empíricamente su consideración a la hora de la resolución de problemas.

Esta fase de modelado, fue la más duradera, debido a las modificaciones del software requeridas. Las pruebas realizadas pasan por ejecutar cada planificador con cada versión modificada del dominio a través de los diferentes problemas de cada dominio para, según el tiempo de ejecución permitido, obtener los resultados de cada uno y poder compararlos entre sí y con los resultados obtenidos de la ejecución previa a la modificación.

Una vez determinada la eficacia de dichas modificaciones sobre diferentes dominios y problemas, se extraerán las conclusiones y las posibles líneas de mejora futuras.

1.2 Objetivos

Como veremos más adelante, los ficheros formados por dominios de planificación contienen objetos, predicados y acciones que se pueden realizar sobre dichos objetos. Estos ficheros, contienen esta información mediante la sintaxis empleada por un lenguaje de especificación llamado PDDL. Mediante la modificación de dichos ficheros, haciendo uso de los diferentes requerimientos y características del lenguaje, vamos a tratar, como objetivo principal del presente proyecto, de estudiar la importancia del modelado y su impacto en la resolución de los problemas de planificación.

Por lo tanto, los subobjetivos del proyecto serían los siguientes:

· Estudio de la Planificación automática y lenguaje de modelado PDDL focalizando en los requisitos del lenguaje aplicables a los diferentes dominios.

· Modificación del dominio seleccionado para las pruebas.

· Realización de pruebas empíricas con diferentes planificadores sobre los dominios y problemas con y sin modelar.

· Evaluación de los resultados obtenidos para obtener las conclusiones sobre el modelado.

1.3 Estructura del documento

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

- "*Introducción y objetivos*", capítulo donde podemos encontrar una visión global del proyecto a modo de introducción y donde se presentan los objetivos del mismo.

- Capítulo 2, el "*Estado de la cuestión*". En este capítulo se presenta una visión general sobre la Planificación Automática, un análisis del lenguaje PDDL y cómo ha sido su evolución a través de las distintas versiones hasta llegar a su estado actual, un esquema de la Competición Internacional de Planificación (IPC) y los dominios implicados en la misma, así como los planificadores según la categoría de la competición donde participaron; todo ello como marco de referencia para la comprensión de este trabajo.

- En el capítulo 3, "Análisis", se introduce el marco teórico subyacente. Se estudian las variaciones del domino en PDDL, así como la selección de los planificadores utilizados.

- En siguiente capítulo es el de "*Pruebas*", dónde se especifican los requisitos de pruebas necesarios y se detalla la batería de pruebas propuesta.

- El capítulo 5, consta de los resultados obtenidos en las pruebas empíricas realizadas, así como la evaluación de dichos resultados.

- Posteriormente, en el capítulo 6, las "*Conclusiones y líneas futuras*", conclusiones obtenidas en el proyecto, resultados y contribuciones. Se proponen también algunas ideas para trabajos futuros, evoluciones del proyecto para conseguir una mejora del mismo.

- El "*Presupuesto",* donde se contempla la estimación del coste del trabajo realizado, se encuentra detallado en el capítulo 7.

- Antes de finalizar, "Anexos", donde se encuentran adjuntos los documentos con las versiones del dominio modificadas para la realización de las pruebas.

- Para concluir, el "Glosario", capítulo que contiene el catálogo con terminología utilizada; y las "Referencias", capítulo donde se detallan las fuentes utilizadas para recabar la información necesaria para la realización del proyecto.

2. Estado de la cuestión

2.1 Acercamiento a la Planificación Automática. Visión General

La planificación es el proceso de desarrollo y producción de planes para el alcance de unos determinados objetivos. La planificación automática, disciplina de la Inteligencia Artificial (IA), estudia este proceso de producción de planes de forma computacional, encontrando y organizando una secuencia de acciones que, anticipándose a los resultados esperados, permitan alcanzar unos objetivos predeterminados.

Los programas que incorporan los algoritmos de planificación, llamados planificadores, consideran tres entradas: Una descripción del estado inicial del mundo, una descripción del objetivo u objetivos a lograr y el conjunto de posibles acciones para lograr dichos objetivos. Cada acción específica generalmente precondiciones para la ejecución de dichas acciones y postcondiciones o efectos, indicando de qué manera se transforma el estado actual del mundo tras su aplicación.

La planificación automática está promovida, en un sentido práctico, por el diseño de herramientas que den acceso a los recursos de manera asequible y eficiente. Además, una motivación teórica es que la planificación es un importante componente del comportamiento racional y uno de los propósitos de la IA es captar computacionalmente los aspectos de la inteligencia, con lo que el estudio de la planificación como razonamiento de la actuación, es un elemento clave para este propósito. Combinando la parte práctica y teórica, la planificación automática impulsa el estudio y diseño de máquinas autónomas inteligentes.

Del mismo modo que existen diferentes tipos de acciones, existen diferentes formas de planificación, teniendo en cuenta la manera en que se percibe y recopila la información, se planifican los movimientos, etc. Siguiendo varios ejemplos:

- Planificación de la ruta y los movimientos, que hace referencia a la elaboración de la trayectoria de un objeto desde una posición inicial a la posición objetivo y el control del camino e intervinientes (camión, grúa, etc…) en ese espacio. Este tipo de planificación está bastante avanzada y tiene en cuenta el modelo de medio ambiente y las restricciones cinemáticas y dinámicas del sistema móvil, ofreciéndonos métodos eficientes de planificar.

- Planificación de la percepción que, por otro lado, tiene que ver con los planes en los que intervienen acciones de detección para recopilar información. Es necesario entonces el modelado e identificación de objetos para detectar el estado actual de un entorno, sabiendo qué información es necesaria, dónde buscarla, qué sensores son adecuados y cómo utilizarlos para cada tarea específica. Por ejemplo, con ello podríamos diseñar un modelo virtual preciso de un escenario urbano. Además, la

recolección de datos es una forma particular de la planificación de la percepción que, en lugar de la detección de la información, se preocupa por la consulta del sistema, pruebas y diagnósticos de los mismos o búsquedas de bases de datos distribuidas en una red.

- Planificación de la navegación donde combinamos los dos problemas anteriores de movimiento y planificación de percepción. Una aplicación sería, por ejemplo, la planificación de rutas para robots móviles en entornos dinámicos. Un robot capaz de superar alteraciones y objetos que aparezcan en su camino llegando a replanificar la ruta en tiempo real.

- Planificación de la manipulación, que hace referencia a la manipulación de objetos. Las acciones deben incluir premisas que impliquen fuerza, tacto, visión y otras informaciones sensoriales. Por ejemplo, para la construcción de ensamblajes, recogiendo objetos por los lados marcados, depositándolos o empujándolos suavemente hasta que encajen en su posición.

- Planificación de la comunicación, donde se tratan los problemas de diálogo y cooperación entre varios agentes. Cómo y cuándo consultar la información y la retroalimentación que debe ser proporcionada.

Existen muchos más tipos de planificación sobre todo en ámbitos sociales y económicos. Como ejemplos, la planificación para el despliegue de medios y la organización de una infraestructura urbana o la planificación financiera y su optimización.

Un enfoque natural para estas diferentes formas de planificación es abordar cada problema con las representaciones y técnicas específicas adaptadas al mismo. Siguiendo una planificación independiente del dominio, particularmente, la planificación de proyectos, en la que los tipos de acciones son principalmente limitaciones temporales y de precedencia; la programación y asignación de recursos, en la que los tipos de acciones incluyen además de los anteriores restricciones en los recursos que usa cada acción lo que nos llevará a la optimización; y la síntesis del Plan, en la cual los modelos de acciones mejoran los anteriores con condiciones necesarias para la aplicación de una acción y efectos de estas.

El modelo conceptual para la planificación y definición de los elementos básicos de un problema se basará en un modelo general de sistema de transición de estados que se representa con una tupla de cuatro elementos: $\sum = (S, A, E, \gamma)$, donde: S es el conjunto finito de estados, A el conjunto finito de acciones, E el conjunto finito de eventos e γ serán las funciones de transición de estado S x A x E. En planificación clásica se considera un modelo alternativo en el que un problema de planificación P se considera una tupla P = (S, A, I, G), donde S y A son el conjunto de estados y acciones, llevando implícita cada acción la transformación producida en el estado. Por

su lado I representa un estado inicial y G un conjunto de metas, en general un estado parcialmente definido. El objetivo de la planificación será por tanto el de encontrar una secuencia de acciones $\{a_1,a_2,\ldots a_n\}$ pertenecientes a A que aplicadas al estado inicial I conduzcan a un estado S_N en el que las metas de G sean ciertas.

2.2 El lenguaje

El logro más importante de la primera competición IPC de 1998 (de la cual hablaremos más adelantes), fue la creación, por Drew McDermott, de un lenguaje común estándar para la definición de los problemas de planificación: PDDL (*Planning Domain Definition Language o* Lenguaje de Definición de Dominios de Planificación), que sigue siendo hoy en día ampliamente utilizado y punto clave para la evaluación comparativa de los planificadores. Este lenguaje se basa en STRIPS (*Stanford Research Institute Problem Solver*) nombre que recibe el lenguaje de entrada al generador de planes automáticos del mismo nombre y ADL (*Action Description Language*), lenguaje de planificación automática particularmente diseñado para robots, considerado una mejora del STRIPS.

Los componentes del lenguaje PDDL son: los tipos de objetos (genéricos) y los objetos, '*types*' y '*Objects*' (cosas que nos interesan del mundo, por ejemplo, un camión o una grúa), predicados '*predicates*' (propiedades de los objetos o relaciones entre ellos, que pueden ser verdaderas o falsas) y funciones 'functions' (que irán apareciendo según evolucione el lenguaje).

Existe la necesidad de representar los estados inicial (dónde empezamos), actual y final del mundo (dónde queremos acabar, cosas que queremos que sean ciertas), donde se presupone un número finito de estados intermedios posibles, acciones deterministas e instantáneas, tiempo implícito y una percepción del mundo aislado, todo lo que no se diga explícitamente que es cierto, es falso. Sobre este 'mundo', se van a realizar un conjunto de posibles acciones que transformaran un estado en otro.

Para elegir un plan, secuencia de acciones que conviertan el estado inicial en final, se seleccionarán las acciones según las precondiciones y postcondiciones/efectos definidas de las mismas. Además, se tendrá en cuenta que, a lo largo de las diferentes versiones y evolución del PPDL se irán añadiendo nuevas funcionalidades y requerimientos.

Las tareas de planificación están separadas en dos ficheros: (1) El fichero del dominio o *domain file*, que contendrá los tipos de objetos, predicados, funciones y las acciones y (2) el fichero del problema o *problem file*, para los objetos, estado inicial y las especificaciones de las metas.

Ambos ficheros tendrían un aspecto similar a este:

Domain file:
(define (domain <nombre-del-dominio>)
(:requirements :definición de requerimientos que acepta el dominio)
(:types <lista-de-tipos>)
<código PDDL para los predicados>
<código PPDL para las funciones>
<código PDDL para la primera acción>
[…]
<código PPDL para la última acción>)

Problem file:
(define (problem <nombre-del-problema>)
(:domain <nombre-del-dominio>)
<código PDDL para los objetos>
<código PDDL para el estado inicial>
<código PDDL para especificaciones del objetivo>
<código PDDL para la métrica>)

A continuación, un ejemplo sencillo de funcionamiento de ficheros de dominio y problema. Para el fichero del dominio, en este ejemplo, no se han incluido tipos *(:types):*

DOMAIN FILE

```
(define (domain <gripper>)
    (:predicates (ROOM ?x) (BALL ?x) (GRIPPER ?x)
                 (at-robby ?x) (at-ball ?x ?y)
                 (free ?x) (carry ?x ?y))
     (:action move :parameters (?x ?y)
                 :precondition (and (ROOM ?x) (ROOM ?y)
                                    (at-robby ?x)
                 :effect           (and (at-robby ?y)
                                        (not (at-robby ?x))))
    (:action pick-up :parameters (?x ?y ?z)
                 :precondition (and (BALL ?x) (ROOM ?y)(GRIPPER ?z)
                                    (at-ball ?x ?y) (at-robby ?y)
                                    (free ?z))
                 :effect          (and (carry ?z ?x)
                                       (not (at-ball ?x ?y)) (not (free ?z))))
    (:action drop :parameters (?x ?y ?z)
                 :precondition  (and (BALL ?x) (ROOM ?y) (GRIPPER ?z)
                                     (carry ?z ?x) (at-robby ?y))
                 :effect          (and (at-ball ?x ?y) (free ?z)
                                       (not (carry ?z ?x))))
                                                                  )
```

Primero se ha definido el dominio <Pinza>, en el que tenemos los predicados (características) del tipo: X es una 'HABITACIÓN', X es una 'BOLA' y X es una 'PINZA'; las propiedades de los mismos 'el robot está en la habitación X', 'está la bola X en el lugar Y', 'está libre el brazo X del robot', 'llevar X a Y' y una serie de acciones como MOVER, COGER y SOLTAR. Para poder ejecutar estas acciones deben cumplirse unas condiciones y una vez ejecutada cada acción tendrá un efecto. En caso de la acción MOVER, que admite los parámetros de entrada X e Y, las precondiciones para su ejecución son que dadas las habitaciones X e Y, el robot estará en la habitación X y al ejecutar la acción MOVER el robot pasará a estar en la habitación Y. Para la acción COGER, con los parámetros de entrada X, Y y Z, las precondiciones nos dicen que la bola X debe estar en la habitación Y, el robot debe estar en la habitación Y también y la pinza Z debe estar libre. En ese caso, la pinza Z deja de estar libre y coge la bola X, que ya no estará en la habitación Y. Para la acción SOLTAR, con los parámetros de entrada X, Y y Z, las precondiciones son que la pinza del robot Z haya cogido la bola X y que el robot esté en la habitación Y; en este caso, el efecto producido es que la bola pasa de estar en X a estar en Y, la pinza del robot vuelve a estar libre y la bola X ya no está en la pinza Z.

PROBLEM FILE

```
(define (problem <gripper-four-balls>)
    (:domain <gripper>)
    (:objects rooma roomb
              ball1 ball2 ball3 ball4
              left right)
    (:init    (ROOM  rooma) (ROOM roomb)
              (BALL ball1) (BALL ball2) (BALL ball3) (BALL ball4)
              (GRIPPER left) (GRIPPER right)
              (at-ball ball1 rooma) (at-ball ball2 rooma)
              (at-ball ball3 rooma) (at-ball ball4 rooma))
    (:goal    (and (at-ball ball1 roomb)
                   (at-ball ball2 roomb)
                   (at-ball ball3 roomb)
                   (at-ball ball4 roomb)))
                                                    )
```

El fichero del problema se ha definido en base al dominio anterior <Pinza>. En él encontramos definidos los objetos habitaciónA y habitaciónB, bolas 1, 2, 3 y 4 e izquierda y derecha. En el estado inicial vemos que habitaciónA y habitaciónB son tipos de 'HABITACIÓN' (del dominio), las bolas 1, 2, 3 y 4 son del tipo 'BOLA' (del dominio) y los objetos izquierda y derecha son del tipo 'PINZA' (del dominio); además, se indica que todas las bolas 1, 2, 3 y 4 se encuentran en la habitaciónA. En cuanto al estado final, se desea que todas las bolas estén en la habitaciónB. Es decir, tenemos un robot, que debe llevar las bolas de la habitaciónA a la habitaciónB.

2.3 Recorrido por las versiones del lenguaje

El lenguaje PPDL ha ido evolucionando y de una versión a otra se han ido introduciendo diferentes características, aunque no todas han sobrevivido y algunas han dejado de utilizarse. A continuación, vamos a realizar un recorrido por las diferentes versiones del lenguaje.

Para definir un estado, lo que necesitamos es un conjunto de objetos o conceptos (paquetes, grúas, localizaciones, tiempo de entrega…) y un conjunto de predicados que relacionen esos objetos (entregar(paquete), en(paquete, localización), …) como ya vimos anteriormente.

Los elementos que intervienen en el dominio y problema, serán introducidos según la definición que hagamos de los mismos en la versión del lenguaje utilizada.

2.3.1 PDDL 1.2

En esta primera versión, se definieron los requerimientos del lenguaje que podrán ser utilizados en el dominio. Los encontraremos definidos de la siguiente manera, con los siguientes calificadores:

:strips	Precondiciones básicas de STRIPS, Añadir *'add'* y borrar '*delete*' por defecto.
:typing	Se permiten tipos (*types*) en la declaración de variables.
:negative-preconditions	Se permite la negación '*not*' en las precondiciones de las acciones.
:disjunctive-preconditions	Se permite la disyunción '*or*' en las precondiciones de las acciones.
:equality	Soporta el = en los predicados.
:existential-preconditions	Se permite la condición de existencia *'exists'* en las precondiciones de las acciones.
:universal-preconditions	Se permite la cláusula '*forall*', para todo, en las precondiciones de las acciones y la descripción del estado final.
:quantified-preconditions	Las dos anteriores juntas (*existential-preconditions* + *universal-preconditions*).
:conditional-effects	Permite la cláusula cuando, '*when*', en los efectos de las acciones, creando efectos condicionales.
:adl	La suma de todas las anteriores.

2.3.2. PDDL 2.1

Esta versión fue creada en 2002 por Maria Fox y Derek Long para la tercera IPC de 2002. Las principales nuevas características que presenta son la introducción de *fluents* (funciones numéricas), acciones durativas para la planificación temporal, métricas para la medición de la calidad y la división del lenguaje en cinco niveles: (1) STRIPS + ADL; (2) *Fluents*; (3) Acciones durativas discretas; (4) Acciones durativas continuas; (5) PPDL 2.1. completo. Solo los tres primeros niveles fueron usados en la competición, los dos siguientes parecen olvidados.

Los f*luents* o coste de las acciones permiten añadir valores numéricos a cada acción, para determinar qué plan tiene el menor coste o qué acción es más adecuada. La nueva etiqueta para este requerimiento es ***:fluents*.** En la definición del dominio añadiremos por tanto un nuevo campo *:functions*, generalmente después de los predicados. El valor se le asignará en el fichero del problema y podrá ser usado en precondiciones de las acciones (junto a los operadores relacionales = > < <= >=) o en los efectos de las mismas (modificando los efectos con las cláusulas incrementar, disminuir, asignar, aumentar proporcionalmente, decrementar proporcionalmente).

El uso de los *fluents* permite opcionalmente definir métricas para medir la calidad del plan, con la etiqueta ***:metric***, para minimizar o maximizar el coste y la premisa coste-total añadida.

Las acciones durativas para la planificación temporal, pueden tener duración y pueden ser paralelas. La nueva etiqueta para este requerimiento es ***:duration***. La duración puede especificarse mediante un valor numérico que se puede calcular usando *fluents*, darse como un intervalo o de duración vacía, lo que nos indicará que la acción durará hasta que una condición previa se convierta en falsa. Para las precondiciones y efectos temporales, es obligatorio el uso de las siguientes etiquetas para las anotaciones temporales: indicando si deben producirse al inicio, '*at star'*, al final, '*at end*' durante todo el tiempo que dura la acción, '*over all'*.

2.3.3. PDDL 2.2

Creada por Stefan Edelkamp, Jörg Hoffmann y Michael Littman en 2004 para la cuarta IPC. Es básicamente igual a PDDL 2.1, en concreto en los 3 primeros niveles, con las novedades de que se desarrollan los predicados derivados (con la etiqueta ***:derived-predicates***), que se pueden automáticamente derivar de la información del problema dado y que no se verán afectados por ningún operador del dominio (lo que en PDDL originalmente se denominaba axiomas) y los literales con tiempo inicial (con la etiqueta *:**timed-initial-literals***), hechos que se volverán verdaderos o falsos en momentos determinados del tiempo que se conocen de antemano, independientemente de las acciones que se ejecuten; se incluyen en el estado inicial junto con el momento en que se volverán ciertos.

2.3.4 PDDL 3.0

Desarrollado por Alfonso Gerevini y Derek Long para la sexta IPC, dentro del ICAPS de 2006. Introduciendo preferencias sobre los objetivos y restricciones sobre la trayectoria de los estados, para focalizar en la calidad del plan y no en su duración, haciendo que el lenguaje sea lo más parecido al mundo real.

Aparecen entonces los objetivos intermedios que se requiere que sean ciertos en ciertos momentos del plan, no solo al final, expresados con operadores modales temporales. La etiqueta para el requerimiento es **:constraints** que se añadirá al dominio pudiendo aparecer en el fichero del dominio o en el fichero del problema. Los operadores de restricción son los que siguen; *always* (cierto todo el tiempo), *sometime* (cierto alguna vez en el plan), *at-most-once* (una vez es cierto, después será falso sin poder volver a ser cierto), *at end* (cierto al final, equivalente a una meta normal), *withint* t (cierto durante la duración del tiempo t), *sometime-before* (un objetivo 'B' será cierto antes de que el objetivo 'A' lo sea) y *sometime-after* (un objetivo 'B' será cierto después de que el objetivo 'A' lo sea), *always-within* t (una vez el objetivo A es cierto, el objetivo B debe ser cierto antes del tiempo t), *hold-during* t1 t2 (cierto entre los tiempos o estados t1 y t2), *hold-after* t (debe ser cierto después de t).

Las restricciones sobre la trayectoria y las metas deben cumplirse para que el plan sea válido. Sin embargo, las preferencias aplicadas a los objetivos y restricciones, de no cumplirse, no hacen que un plan sea inválido. Se añade el requerimiento con la etiqueta **:preferences**, que puede aparecer en el fichero de dominio en cuyo caso se aplicará a todo el problema en el fichero del problema aplicándose entonces a objetivos o restricciones concretas.

Este concepto de preferencias también puede aplicarse a las precondiciones, diciendo que es preferible que se cumpla determinada precondición, pero no es obligatorio. Existen penalizaciones asociadas a cada preferencia para llevar la cuenta de las preferencias que no han sido satisfechas. Estas, se pueden utilizar en la métrica del problema.

2.3.5 PDDL 3.1

Esta versión fue la oficial de la parte determinista de la sexta y séptima IPC en 2008 y 2011. Como novedad se añade el concepto de *object fluents* de manera análoga a los *fluents* numéricos de PPDL2.1, que hoy en día se conocen como ***:action-costs***. Con este requisito, se permite introducir funciones (costes en las acciones), que pueden ser utilizadas tanto en las precondiciones de las acciones como en los efectos de las mismas.

Un ejemplo de definición de funciones en el fichero del dominio sería el siguiente:

(:functions
(length ?rooma – room ?roomb – room)
(total-cost)
)
[Se definen las funciones longitud entre dos habitaciones y coste total]

Cuando las funciones son usadas en precondiciones, se acompañan de los operadores relacionales =, >, <, >= y <=, mientras que, si son usadas en los efectos, los valores de las mismas son modificados mediantes *increase, decrease, assign, scale-up, scale-down.*

· Ejemplo de uso en precondiciones:

(< *(length ?x ?y) 10)*
[Longitud entre x e y que sea menor de 10]

· Ejemplo de uso en efectos:

(increase (total-cost) (length ?x ?y))
[El *coste total de la acción es la longitud entre las dos habitaciones.]*

En la actualidad, existen pocos planificadores que manejen precondiciones numéricas.

2.3.6 PPDDL

Con esta versión se crea una nueva descripción del dominio del lenguaje de planificación llamada PPDDL 1.0, "*Probabilistic Planning Domain Definition Language*", que nos va a permitir modelar los problemas de planificación probabilísticos. Fue utilizado como el idioma de entrada para la categoría probabilística de la cuarta IPC. PPDDL se inspira en PDDL2.1 (Fox & Long, 2003), el lenguaje de dominio-descripción para dominios deterministas utilizados en la IPC-3.

El objetivo fue unificar dos comunidades de investigación para crear herramientas comparables comunes. La comunidad de investigadores de la toma de decisiones basada en modelos de Markov (MDP) y la de investigadores que incorporan los conceptos teóricos probabilísticos en los algoritmos de planificación.

2.4 Estado actual

La última versión del lenguaje es la PDDL 3.1. Aunque existen en la actualidad otros sucesores, además de la versión PPDDL, variantes y extensiones del lenguaje PDDL en los que no profundizaremos, tales como, PDDL+ (extensión de PDDL2.1 aproximadamente entre 2002 y 2006), NDDL (*New Domain Definition Language,* Nuevo lenguaje de definición de dominio, es la respuesta de la NASA para PDDL alrededor de 2002), MAPL (Multi-Agent Planning Language, Lenguaje de la planificación multi-agente, extensión de PDDL2.1 surgida alrededor de 2003), OPT (Ontology with Polymorphic Types, Ontología con tipos polimórficos, que fue una profunda ampliación de PDDL2.1 por Drew McDermott entre 2003-2005, con algunas similitudes con PDDL +) , APPL (Abstract Plan Preparation Language, Lenguaje de preparación del plan abstracto, variante más reciente de NDDL desde 2006, que es más abstracto que la mayoría de lenguajes de planificación existentes, como PDDL o NDDL), RDDL (Relational Dynamic influence Diagram Language, idioma oficial de la categoría de incertidumbre de la séptima IPC en 2011) o MA-PDDL (Multi Agent PDDL, extensión minimalista, modular del PDDL3.1 que se introdujo en 2012).

Nosotros nos centraremos en la versión PDDL 2.2 que es la que implementan la mayoría de los planificadores.

En el apartado de anexos, se ha incluido la definición completa de sintaxis BNF* del lenguaje PDDL 3.1 traducida, con la descripción del dominio, problema y requerimientos. Publicación original de Daniel L. Kovacs en 'Complete BNF description of PDDL 3.1(completely corrected)' en 2011.

*) Backus-Naur Form: metalenguaje para expresar gramáticas libres de contexto, es decir, manera formal de describir lenguajes formales.

2.5 La Competición Internacional de Planificación

Desde la primera edición en 1998, bienalmente al principio y cada tres años en la actualidad tiene lugar la Competición Internacional de Planificación (IPC – *International Planning Competition*), en el marco de la Conferencia Internacional sobre Planificación y Programación Automática (ICAPS - *International Conference on Automated Planning and Scheduling*), siendo piedra clave en la comunidad de investigación de planificación en todo el mundo.

Los principales objetivos de la IPC son: crear nuevos dominios de aplicación, recopilando tantos datos como sea posible para ponerlos a disposición de la comunidad, proponiendo nuevas direcciones para la investigación y suministrando un conjunto de problemas comunes y un formalismo de representación para ayudar a la comparación y evaluación empírica de los sistemas de planificación y su rendimiento mediante algunas métricas específicas.

Actualmente, la organización de la competición comprende tres partes o categorías: (1) Una categoría para la evaluación de los planificadores independientes del dominio en entornos deterministas y completamente observables; (2) una categoría de planificación y aprendizaje para los planificadores que son capaces de aprender y explotar el conocimiento específico del dominio en la planificación determinista; y (3) una categoría para los planificadores independientes del dominio capaces de planificar en condiciones de incertidumbre (continua o discreta).

Centrándonos en la competición del 2011, que será el marco en que se desarrollará este proyecto, los organizadores para las categorías anteriormente mencionadas fueron: Ángel García Olaya, Carlos Linares López y Sergio Jiménez (Determinista); Scott Sanner y Sungwook Yoon (Incertidumbre); Sergio Jiménez, Amanda Coles y Andrew Coles (Aprendizaje).

En su parte determinista, se basó en PDDL 3.1 (STRIPS + *action-costs*, temporal STRIPS, aunque finalmente dado que la mayor parte de los planificadores no soportaban todo el conjunto de PDDL 3.1, los dominios se codificaron con un conjunto restringido de las características de PDDL 2.2.

Más concretamente vamos a revisar la categoría determinista. Esta se divide en tres bloques principales: planificación secuencial, que tiene el objetivo de minimizar el coste total de los planes; temporal, para minimizar el *makespan* (tiempo entre el inicio y el fin del plan) y preferencias, plan con penalización mínima. Cada parte a su vez se divide en dos subgrupos: *satisficing* (es suficiente encontrar un plan para considerar un problema resuelto, independientemente de la calidad de dicho plan) y planificación óptima (se debe encontrar el plan de mínimo coste, cualquier otro plan es considerado no válido). Además, a partir de 2011, la IPC cuenta para el bloque de *satisficing* secuencial con un subbloque de multi-core, en el que se pueden usar todos los procesadores de un equipo (en el resto de categorías solamente uno de los

procesadores puede ser usado).

En ese año la competición contó con el siguiente número de sistemas de planificación: 27 de satisficing secuencial, 12 de óptima secuencial, 8 de satisficing temporal, 7 de satisficing secuencial multi-core, 11 de incertidumbre y 8 de aprendizaje.

Para la evaluación, a los competidores se les dio un conjunto de problemas para probar sus planificadores en sus propias máquinas, presentando una versión final de código ejecutable que los organizadores ejecutaron con todos los dominios y problemas informando posteriormente de los resultados.

Remarcar la importancia de realizar planes de buena calidad que, dependiendo de cada categoría, obtendrán mejor puntuación sin hacer tanto énfasis en el tiempo de solución. Por ejemplo, para la parte de optimización, cada planificador debe resolver cada problema en un tiempo máximo de 30 minutos, no teniéndose en cuenta el tiempo concreto siempre que se resuelva dentro de esos 30 minutos; Si algún planificador produce algún plan no óptimo, el resto de resultados de ese planificador en ese dominio serán ignorados, aunque dando a los participantes la oportunidad de corregir el error. En definitiva, se evalúa el número de problemas resuelto contando con que todos los problemas son equivalentes (no se tiene en cuenta que unos sean más sencillos que otros). La puntuación más alta gana.

A continuación, se muestra una figura resumen con la Historia de la Competición Internacional de Planificación (IPC), publicada por la revista IA MAGAZINE en el artículo "A Survey of the Seventh International Planning Competition" de 2012.

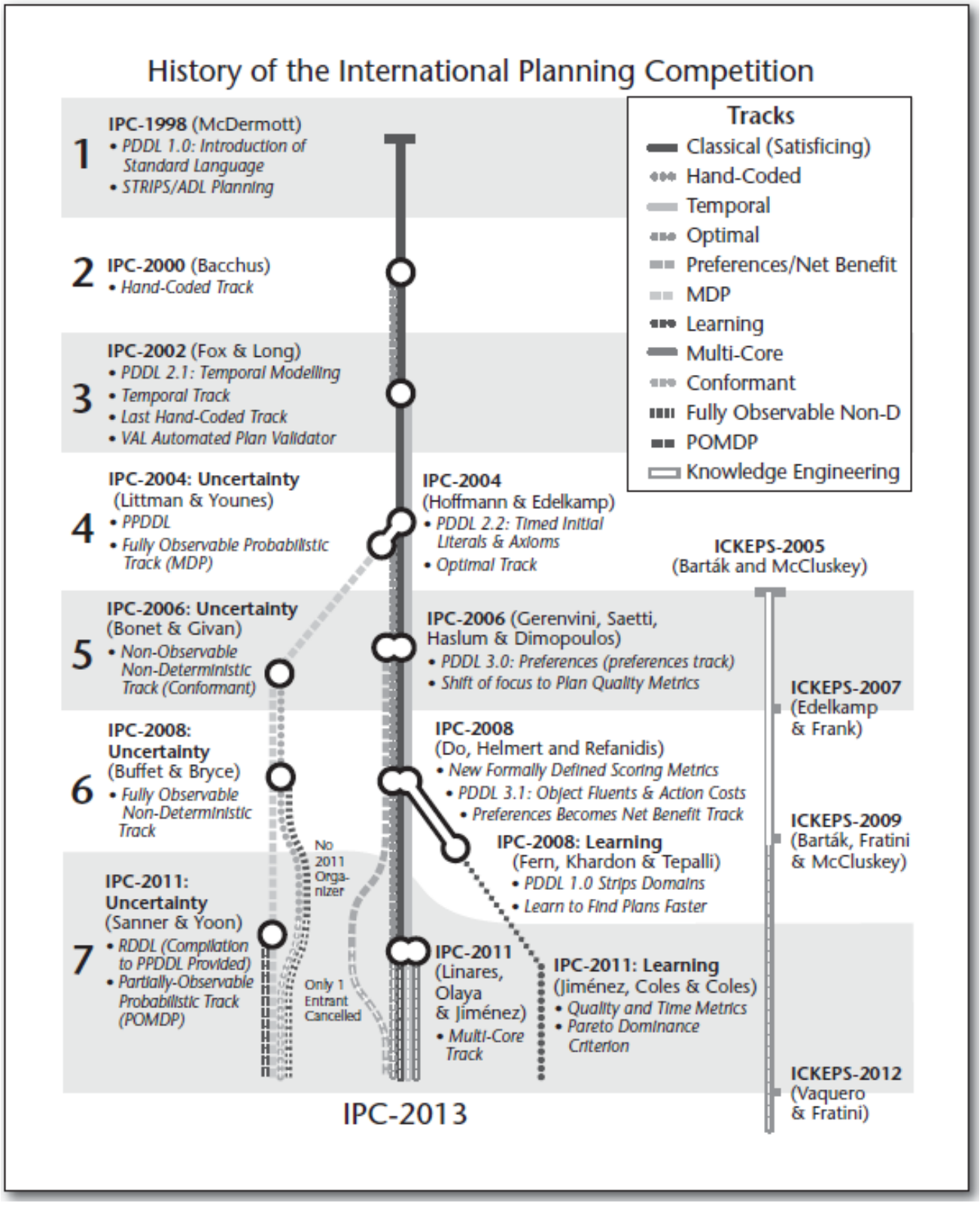

Figura 1. Historia de la Competición Internacional de Planificación (IPC).

2.6 Dominios de la competición

A continuación, vamos a ver una descripción de los dominios utilizados en la competición IPC de 2011, para las partes de satisfacing, óptima y multi-core. Vamos a ver las características de cada uno para un conjunto de problemas seleccionados. Para que sea posible la justa comparación entre diferentes partes de la competición, se utilizan los mismos problemas para multi-core secuencial y satisficing secuencial.

DOMINIO	VERSIÓN PPDL	ORIGEN	CATEGORÍA
Barman	STRIPS	Nuevo	Secuencial
Elevators	action-costs*	IPC 2008	Secuencial, temporal
Floortile	action-costs	Nuevo	Secuencial, temporal
Nomystery	action-costs	Nuevo	Secuencial
Openstacks	action-costs	IPC 2008	Secuencial, temporal
Parcprinter	action-costs	IPC 2008	Secuencial, temporal
Parking	action-costs	IPC 2008	Secuencial, temporal
Pegsol	action-costs	IPC 2008	Secuencial, temporal
Scanalyzer	action-costs	IPC 2008	Secuencial
Sokoban	action-costs	IPC 2008	Secuencial, temporal
Tidybot	STRIPS	Nuevo	Secuencial
Transport	action-costs*	IPC 2008	Secuencial
Visit-all	STRIPS	Nuevo	Secuencial
Woodworking	action-costs*	IPC 2008	Secuencial

**Se definen otros fluents aparte del coste total del plan.*

***Tabla 1**. Cuadro resumen dominios IPC 2011*

2.6.1 Barman (Camarero)

Este dominio fue creado para la competición IPC 2011 por Sergio Jiménez Celorrio. En él, hay un robot-camarero que manipula un conjunto de dispensadores de bebidas, vasos y una coctelera. El objetivo es encontrar el plan de acciones para que el robot sirva un número de bebidas o cócteles que pueden contener uno o varios ingredientes, sabiendo que el robot sólo puede coger un objeto a la vez y que los vasos deben estar vacíos y limpios para ser llenados. Todas las acciones tienen el mismo coste. El conjunto de problemas para este dominio se genera aumentando el número de bebidas a servir.

2.6.2 Elevators (Ascensores)

Este dominio llegó a través de la IPC-2008 inspirado por otro dominio, Miconic de la IPC-2000. Sin embargo, su autor Ioannis Refanidis lo volvió a diseñar desde cero.

Simula el trabajo de un conjunto de ascensores que necesitan transportar a un número de personas de una posición inicial a un destino ambos conocidos, a través de N + 1 pisos, numerados de 0 a N. El edificio se divide en bloques iguales de M + 1 plantas y bloques adyacentes tienen un piso en común. Un ejemplo podría ser N = 12 y M = 4, entonces tendríamos 13 pisos (que van del 0 a 12), que forman 3 bloques de 5 pisos cada uno, siendo estos pisos por bloque 0-4, 4-8 y 8-12.

Existen dos tipos de ascensores, K ascensores rápidos que realizan una única parada en M/2 plantas y tienen capacidad de X pasajeros; y L ascensores lentos que realizan paradas en cada piso teniendo una capacidad de Y pasajeros (donde usualmente Y<X). Además, existen costes asociados a cada ascensor por partida/parada y movimiento. Los ascensores rápidos tienen mayor coste en movimiento e insignificante en inicio y parada, mientras que los ascensores lentos es al contrario.

Las versiones de este dominio son las siguientes:

-Secuencial: La función objetivo es reducir al mínimo el coste total de movimiento de los pasajeros a sus destinos. El coste total se incrementa cada vez que un ascensor se pone en marcha, detiene o se mueve.

-Temporal: Cada acción tiene una duración. El objetivo es minimizar el *makespan* del plan.

-Temporal numérico: Al igual que en la versión Temporal; se han utilizado *fluents* numéricos para representar las plantas y las capacidades.

-Óptima o Beneficio neto: Llevar a los pasajeros a sus destinos es un objetivo no obligatorio. Hay una penalización asociada con cualquier pasajero que no esté en su piso de destino. La sanción es una función de la diferencia entre el origen y el piso de destino (que supone para el pasajero el uso de las escaleras). La función objetivo es reducir al mínimo el coste total, expresado en función del coste de mover los ascensores y el coste de no hacer llegar a los pasajeros a sus destinos.

-Beneficio neto - numérico: Similar al anterior, beneficio neto, pero utilizando *fluents* numéricos para representar los pisos y las limitaciones de capacidad.

2.6.3 Floortile (Baldosa)

Tomás de la Rosa creó este dominio para la IPC-2011. En él, un conjunto de robots utiliza diferentes colores para pintar patrones en las baldosas de un piso de m x n.

Los robots pueden moverse en cuatro direcciones: arriba, abajo, izquierda y derecha, pintando de un solo color a la vez, aunque pueden cambiar sus pistolas de pulverización a cualquier color que esté disponible. Sin embargo, solo podrán pintar las baldosas que están delante (hacia arriba) o detrás (hacia abajo) y una vez que un azulejo ha sido pintado ningún otro robot puede pasar sobre él.

Para el conjunto de prueba de la IPC-2011, los robots tienen que pintar una cuadrícula con un modelo en blanco y negro alternando siempre el color de la celda formando una especie de tablero de ajedrez. En cada problema, se ha ido aumentando el tamaño de la cuadrícula y el número de robots de modo que los problemas más difíciles consistían en una cuadrícula de 7 × 6 con 3 robots.

Las acciones permitidas para pintar azulejos, cambiar el color de la pistola y para moverse, tiene un coste diferente. Nótese por ejemplo que ascender, es más costoso que moverse en cualquier otra dirección.

2.6.4 Nomystery (Sin misterio)

Nomystery es un dominio de transporte, muy similar al dominio 'Transporte', diseñado por Jorg Hoffmann y Hootan Nakhost. Fue incluido en la IPC-2008 e IPC-2011 para estudiar la planificación de recursos limitados. En él, un camión se mueve en un grafo ponderado y debe transportar un conjunto de paquetes entre los nodos.

El generador crea primero un grafo no dirigido conectado al azar con N nodos y agrega K paquetes con orígenes y destinos aleatorios. Los pesos de las aristas se distribuyen de manera uniforme entre 1 y un número entero W. Se calcula la cantidad mínima necesaria de combustible M y el suministro inicial de combustible, ajustado en [C × M], donde C ≥1 es un parámetro de entrada del generador. Los problemas aumentan en dificultad cuando C es más cercano a 1, dado que si C = 1 sólo el plan óptimo resolvería el problema. Las acciones de moverse a lo largo de los bordes y la carga / descarga de paquetes tienen un consumo de combustible, surgiendo la necesidad de economizar el recurso limitado.

Es un dominio sencillo para el estudio del comportamiento de algoritmos de planificación en problemas de planificación de recursos.

2.6.5 Openstacks (Openstacks)

Este dominio se introdujo en la IPC-2006 por primera vez y fue reutilizado posteriormente en la IPC-2008 e IPC-2011. Está basado en el problema de optimización combinatoria 'open stacks mínimo máximo', que se podría resumir de la siguiente manera: "Un fabricante tiene una serie de pedidos, cada uno con una combinación de diferentes productos. Sólo se puede fabricar un producto a la vez, pero la cantidad total requerida de ese producto se hace toda al mismo tiempo (porque el cambio de hacer un producto a otro requiere una parada de producción). En el momento en que el primer producto solicitado en un pedido se ha fabricado, a la hora, todos los productos incluidos en ese pedido se habrán fabricado también. Mientras tanto, la orden se dice que esta 'abierta' y durante este tiempo se requiere una pila (un espacio de almacenamiento temporal). El problema es ordenar la fabricación de los diferentes productos de manera que el número máximo de pilas que estén en uso simultáneamente sea menor, es decir, el número de pedidos que se encuentran en producción simultánea, se reduzca al mínimo".

Este es un problema bastante complejo ya que, aunque la búsqueda de un plan es bastante trivial, encontrar un plan de alta calidad es difícil.

Se generaron dos versiones de este dominio en la IPC-2008, STRIPS y ADL. Pero dado que hay pocos planificadores compatibles con ADL, en la IPC-2011 se utilizó únicamente la versión STRIPS. El objetivo es reducir al mínimo el número de pilas totales, representadas usando predicados y precondiciones universalmente cuantificadas. Los objetivos son igual al número de pedidos y existen tantos productos como pedidos, pero cada producto puede utilizarse en más de un pedido. Además, el número de productos que componen un pedido es diferente en cada uno. Para escalar en dificultad, tanto el número de pedidos como la cantidad de productos en de cada pedido va en aumento.

2.6.6 Parcprinter (Impresora Parc)

Este es un dominio sobre los modelos de funcionamiento de una impresora multi-motor, desarrollado en el Centro de Investigación de Palo Alto (*Palo Alto Research Center - PARC*). La compilación en PDDL fue realizada por Rong Zhou y fue utilizado por primera vez en la IPC-2008. Según los autores, representa el primer éxito de aplicación industrial de la planificación temporal independiente del dominio.

"Este tipo de impresora puede manejar varios trabajos de impresión al mismo tiempo. Hojas múltiples, que pertenecen al mismo trabajo o trabajos diferentes, se pueden imprimir simultáneamente usando múltiples IMEs (*'Image Marking Engines')* o motores de marcado de imagen. Cada IME puede ser o bien de color, que puede imprimir tanto imágenes en color como en blanco y negro, o bien mono, que sólo puede imprimir imágenes en blanco y negro. Cada hoja tiene que pasar a través de diferentes componentes de la impresora, tales como el alimentador, transportador, IME, el

inversor o la unidad de acabado, siendo necesaria la llegada a la unidad de acabado estando el trabajo en orden. Por lo tanto, la hoja (n + 1) tiene que ser apilada en la misma unidad de acabado que la hoja n del mismo trabajo, teniendo que llegar inmediatamente después de la hoja n a la unidad, es decir, sin otras hojas apiladas entre esas dos hojas consecutivas.

Teniendo en cuenta que los IMEs son heterogéneos (mezcla de color y mono) y pueden funcionar a diferentes velocidades, la optimización de las operaciones de esta impresora para una mezcla de trabajos de impresión, cada uno de ellos con una mezcla arbitraria de páginas a color/B&N y ya sean simples (impresión de un solo lado) o dúplex (impresión a doble cara), es un problema complicado".

Se modelaron 3 impresoras: 2 de ellas con 2 IMEs de color y blanco y negro y una tercera con 4 IMEs, dos de cada clase. La impresión de las hojas puede realizarse por uno o ambos lados. Cada acción lleva asociado un coste diferente siendo la impresión con un IME a color más caro que el uso de uno blanco y negro. Inicialmente todas las hojas están en blanco y para llegar al estado final cada hoja tendrá que ser impresa ya sea en el frente, en la parte posterior o ambas y con la imagen a color o en blanco y negro. En los objetivos se describe también de qué lado tiene que estar cada hoja y la bandeja de acabado donde tiene que ser apilada.

En el conjunto de problemas de este dominio, lo que se pretende es maximizar la productividad de la impresora, lo que equivale a terminar de imprimir todas las solicitudes de trabajos de impresión lo más rápido posible. Para reducir la dificultad de los problemas, se crean solicitudes de impresión de un solo puesto de trabajo con varias hojas que se establecen al azar (impresión de un solo lado o simple, impresión a doble cara o dúplex) y cada imagen también se selecciona al azar (mono o color). El número de hojas varía de 1 a 20.

2.6.7 Parking (Estacionamiento)

Este dominio fue introducido por primera vez por los organizadores de la categoría de aprendizaje de la IPC-2008. Trata del estacionamiento de los coches en una calle, en la cual pueden ser aparcados en doble fila, pero no triple fila. El propósito es encontrar un plan para pasar de una configuración de coches estacionados a otra.

El conjunto de problemas contiene 2 * (N-1) coches, permitiendo un espacio libre en la acera y garantizando la resolución. Todas las acciones tienen el mismo coste.

2.6.8 Pegsol (Solitario Peg)

Como ocurre en otros dominios, este se basó en un conocido juego: *Peg solitaire*. "La princesa de Soubise jugando al solitario, 1687, Peg Solitaire (o Solo Noble) es un juego de mesa para un jugador que implica el movimiento de clavijas en un tablero con agujeros. Algunos conjuntos utilizan canicas en un tablero con hendiduras. El juego es conocido simplemente como solitario en el Reino Unido, o también se conoce como Brainvita (especialmente en la India).
[..]
El juego estándar llena todo el tablero con clavijas excepto por el agujero central. El objetivo es, haciendo movimientos válidos, vaciar todo el tablero a excepción de una clavija aislada que queda en el agujero central."

Figura 2. Juego de Solitario Peg

Una clavija puede moverse saltando por encima de otra adyacente horizontal o verticalmente a un agujero vacío adyacente a la clavija saltada. Se retira entonces la clavija que acabamos de saltar. La única acción con coste es iniciar un movimiento.

Para la competición de 2011 los problemas se tomaron de la IPC 2008. Se generaron un total de 105 problemas (algunos de ellos sin solución), concluyendo de la competición anterior que es el número de clavijas y no el número de movimientos requeridos lo que hace difícil para los planificadores encontrar una solución. Este dominio fue incluido para comprobar que los planificadores de ese año eran, al menos, tan buenos como los de la competición pasada.

2.6.9 Scanalyzer (Escáner analizador manual)

Malte Helmert, Hauke Lasinger y Robert Mattmuller crearon este dominio para la IPC-2008. En él se modela el problema de la gestión logística de un invernadero automatizado que, estructuralmente, es un problema de permutación que tiene semejanzas con otros problemas como el Cubo de Rubik o el Top-spin, pero que tiene una aplicación real.

Se modela la plataforma scanalyzer LemnaTec, un sistema de cintas transportadoras diseñadas para recopilar automáticamente datos e imágenes visibles e infrarrojas de fluorescencia, de cultivos dentro de uno o varios invernaderos. De esta manera se permite construir un invernadero inteligente para buscar la forma de aumentar el crecimiento y la resistencia de los cultivos.

El dominio ha sido diseñado de manera que existen una serie de cintas paralelas donde se colocan una serie de lotes de plantas. El contenido de las cintas se puede intercambiar utilizando un sistema interior de correas. Hay también una cámara de imágenes donde las plantas son analizadas y este ciclo de análisis comprende que las plantan que comienzan en una cinta, pasen a través de la cámara y terminen en una cinta diferente. Existen las acciones para girar lotes y analizarlos, siendo la segunda acción más costosa que la primera. El objetivo es que todos los lotes de plantas sean analizados.

La complejidad de los problemas aumenta con el aumento del número de cintas y lotes.

2.6.10 Sokoban (Sokoban)

Este dominio está basado en un famoso juego: "Sokoban (guardián de almacén), es un tipo de rompecabezas de transporte, en el que el jugador empuja las cajas o cajones en el interior de un almacén, tratando de conseguir que queden situadas en los lugares de almacenamiento. El rompecabezas se implementa normalmente como un videojuego."

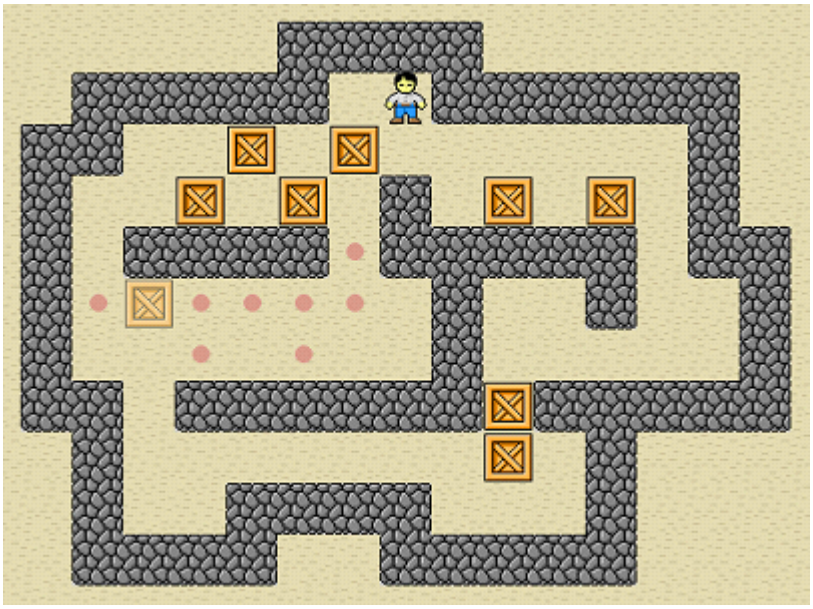

Figura 3. Juego Sokoban

Para el conjunto de prueba de la IPC-2011, en vez de generar nuevos problemas se han reutilizados los problemas más complejos de la IPC anterior del 2008, ya que el rendimiento de los planificadores era pobre. Para esta competición sólo las acciones de movimiento tienen coste, siendo el número de cajas el número de objetivos. Se aumenta la complejidad aumentando el número de cajas y en algunos casos las cajas deben ser trasladadas a posiciones temporales para permitir que otras cajas lleguen a sus lugares finales.

2.6.11 Tidybot (Robot de limpieza)

Este dominio, del autor Bhaskara Marthi, trata sobre las tareas de limpieza del hogar donde uno o más robots deben recoger un conjunto de objetos y ponerlos en sus lugares de destino. El dominio simplifica una versión de una tarea real y está inspirado en el robot PR2 desarrollado en Willow Garage.

El mundo tiene una estructura de cuadrícula en 2D con ubicaciones navegables y superficies en las que pueden encontrase los objetos. El objetivo es transportar los objetos de sus superficies originales a sus superficies destino.

Para ello, los robots tienen un dispositivo de agarre o pinza que se moverá dentro de un radio máximo, mientras el robot esté estacionado. Solo podrán transportar un objeto a la vez en la pinza, aunque también pueden hacer uso de un carro que puede contener varios objetos. Todas las acciones tienen el mismo coste.

El generador de problemas crea mundos que contienen superficies rectangulares (tablas) y recintos en forma de U (armarios) que serán las posiciones finales de los objetos. Del mismo modo, los objetos dentro de los armarios solo pueden cogerse a través de la puerta (la abertura en forma de T). Los robots pueden alcanzar objetos de las mesas desde cualquier lugar de la misma.

2.6.12 Transport (Transporte)

El dominio 'Transporte' fue creado por Malte Helmert y Matthias Westphal para la IPC-2008 y es una variante (más interesante en términos de planes de trabajo y las estructuras de costes) de los dominios de logística ampliamente utilizados en la planificación automática. En él, una serie de camiones tienen que recoger y entregar paquetes en diferentes lugares dentro de una ciudad, representada por una cuadrícula de nodos. Cada camión, podrá transportar simultáneamente un número de paquetes en función a su capacidad y el movimiento tiene un coste diferente dependiendo de la longitud de la carretera. Recoger o dejar un paquete cuesta 1. A diferencia del dominio NOMYSTERY, no hay ninguna restricción sobre el uso de combustible.

En el conjunto de problemas, se aumenta la dificultad con el número de paquetes, camiones y ubicaciones dentro de la ciudad. En la IPC-2011, los problemas no contenían más de 3 ciudades.

2.6.13 *VisitAll (VisitAll)*

Nir Lipovetzky y Héctor Geffner crearon este dominio para evaluar el desempeño de los planificadores en problemas con objetivos contradictorios. Está inspirado en el dominio '*Dispose*' creado por Héctor Palacios y Héctor Geffner para la IPC-2008.

En el dominio, un robot situado en el centro de una cuadrícula de n x n debe visitar todas las celdas de la cuadrícula. Solo existe una acción, que es mover el robot de una celda a otra adyacente.

Hacer hincapié en que los dominios como Visit-All muestran que algunos de los problemas que son difíciles para los planificadores de búsquedas heurísticas 'puras', no son difíciles en absoluto, no son más que el resultado de múltiples objetivos fáciles pero contradictorios que a menudo se pueden lograr trivialmente, una vez que son serializados.

2.6.14 *Woodworking (Tratamiento de la madera)*

Malte Helmert y Gabrielle Roger introdujeron este dominio por primera vez en la IPC-2008. Es una variante del dominio '*Schedule*' de la IPC-2000, centrándose en los costes de las acciones y la interacción entre los objetos. Se simulan las obras en un taller de carpintería, donde existe cierta cantidad de listones de madera que tienen que ser cortados, pulidos, coloreados, etc., usando diferentes herramientas con diferentes costes. El objetivo es producir una serie de piezas.

La dificultad de los problemas aumenta cuanto mayor sea el número de piezas y listones iniciales. Sin embargo, el efecto de la cantidad de herramientas en la dificultad es menos predecible.

2.7 Planificadores

En la IPC de 2011, participaron un total de 55 planificadores agrupados en 31 equipos, representando a 11 países diferentes (Australia, Canadá, China, Francia, Alemania, India, Israel, Italia, España, Reino Unido y EE.UU). Algunos de estos planificadores eran versiones del mismo planificador adaptadas para las diferentes categorías de la competición.

Cada planificador, tenía que tener en cuenta las siguientes limitaciones para la competición, a la hora de resolver los problemas: Hasta 1800 segundos (tiempo máximo 30 min.), 6 GB de memoria RAM y 750 GB de disco duro.

Se anunciaron cuatro categorías de planificación y otras tres más estaban abiertas a la participación (óptima temporal, *satisficing* preferencias y óptima preferencias) pero debido al escaso número de planificadores presentados para ellas finalmente no fue posible incluirlas. Nos centraremos en los planificadores según la categoría o bloque en el que participan.

2.7.1 Planificadores de la categoría satisficing secuencial

El objetivo de los planificadores de esta categoría es minimizar el coste de las acciones (suma de *action-costs);* basándose en STRIPS + *action costs (+ fluents).*

2.7.1.1 Listado de planificadores

A continuación, se muestra un listado de los 27 planificadores participantes y de los autores de los mismos.

PLANIFICADOR	AUTORES
ACOPlan	Marco Baioletti, Alfredo Milani, Valentina Poggioni, Fabio Rossi
ACOPlan2	Marco Baioletti, Alfredo Milani, Valentina Poggioni, Fabio Rossi
Arvand	Hootan Nakhost, Martin Mueller, Richard Valenzano, Fan Xie
BRT	Vidal Alcázar, Manuela Veloso
CBP	Raquel Fuentetaja
CBP2	Raquel Fuentetaja
Roamer	Qiang Lu, You Xu, Ruoyun Huang, Yixin Chen
CPT4	Vincent Vidal
DAE-YAHSP	Johann Dréo, Marc Schoenauer, Pierre Saveant, Vincent Vidal
Fast Downward Autotune-1	Chris Fawcett, Malte Helmert, Holger Hoos, Erez Karpas, Gabi Röger, Jendrik Seipp
Fast Downward Autotune-2	Chris Fawcett, Malte Helmert, Holger Hoos, Erez Karpas, Gabi Röger, Jendrik Seipp
Fast Downward Stone Soup-1	Malte Helmert, Erez Karpas, Silvia Richter, Gabi Röger, Jendrik Seipp
Fast Downward Stone Soup-2	Malte Helmert, Erez Karpas, Silvia Richter, Gabi Röger, Jendrik Seipp
Fork Uniform	Michael Katz, Carmel Domshlak
LAMA-2008	Silvia Richter, Matthias Westphal
LAMA-2011	Silvia Richter, Matthias Westphal, Malte Helmert, Gabi Röger
Lamar	Alan Olsen, Daniel Bryce
LPRPG-P	Amanda Coles, Andrew Coles
Madagascar	Jussi Rintanen
Madagascar-p	Jussi Rintanen
POPF2	Amanda Coles, Andrew Coles, Maria Fox, Derek Long
Probe	Nir Lipovetzky, Héctor Geffner
Randward	Alan Olsen, Daniel Bryce
SATPLANLM-C	Dunbo Cai, Minghao Yin
Sharaabi	Bharat Ranjan Kavuluri
YAHSP2	Vincent Vidal
YAHSP2-MT	Vincent Vidal

Tabla 2 – *Listado de planificadores: Satisficing Secuencial*

2.7.1.2 Resultados obtenidos

La siguiente figura nos muestra un resumen de los resultados obtenidos en esta parte de la competición por los planificadores para cada dominio. En la columna de la izquierda y de arriba a abajo la lista de planificadores de mayor a menor puntuación. Las columnas de la figura muestran los dominios y un código de colores en gamas de gris según la cantidad de problemas resueltos (más oscuro significa más problemas resueltos en ese dominio).

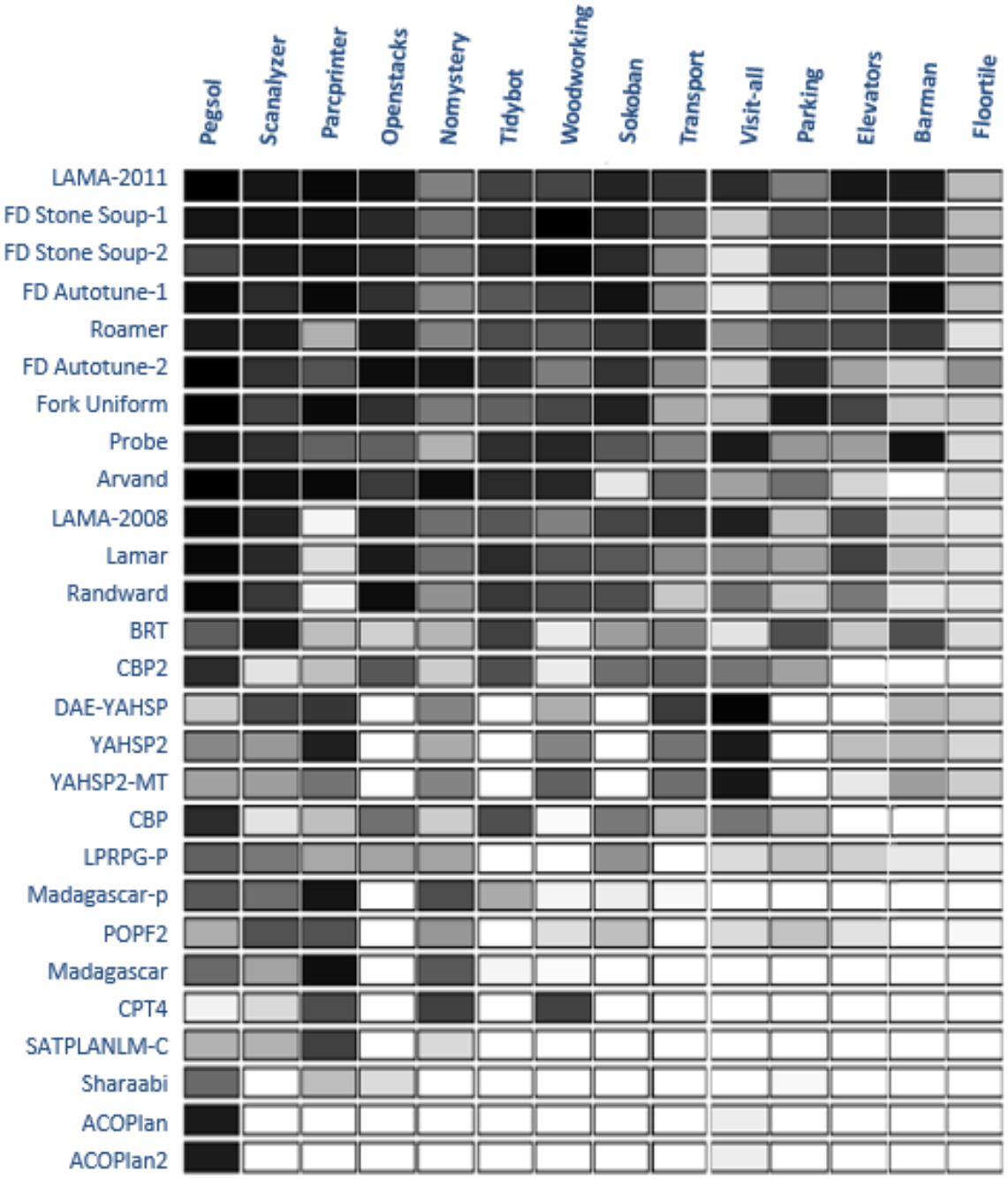

Figura 4 - Resultados categoría satisficing secuencial

Según los puntos obtenidos en función de los problemas resueltos por cada planificador hay un claro ganador, LAMA 2011. Seguido de cerca por el subcampeón Fast Downward Stone Soup-1.

2.7.1.3 Planificadores a destacar

2.7.1.3.1 LAMA

LAMA es un planificador con un sistema proposicional enfocado en la búsqueda heurística con hitos. Existen dos versiones de LAMA, la original desarrollada para la IPC de 2008 y una que se introdujo en la IPC 2011.

El planificador consta de tres programas distintos: 1 El traductor (escrito en Python), 2 El módulo de recopilación del conocimiento (escrito en C ++) y 3 El motor de búsqueda (también escrito en C ++). Para resolver una tarea de planificación, los tres programas se llaman en secuencia comunicándose a través de archivos de texto. Los planes de calidad que devuelve son secuenciales.

2.7.1.3.2 FD Stone soup

Fast Downward Stone Soup es un portfolio de planificadores secuenciales que utiliza diversas heurísticas y algoritmos de búsqueda.

El planificador basa su sistema de planificación en dos observaciones: 1 No existe un único algoritmo de búsqueda común y heurística que domine a todos los demás para la planificación clásica; 2 La cobertura de un algoritmo de planificación no aumenta significativamente cuando le das más tiempo de ejecución, es decir, si un planificador no resuelve una tarea de planificación rápidamente, es probable que no la resuelva nunca.

2.7.2 Planificadores de la categoría óptima secuencial

El objetivo es el mismo que en la categoría satisficing secuencial, pero en este caso solamente se aceptan como válidos los planes óptimos (de mínimo coste) para cada problema.

2.7.2.1 Listado de planificadores

Hubo un total de 12 planificadores participantes que listamos a continuación, junto a los autores de los mismos.

PLANIFICADOR	AUTORES
BJOLP	Erez Karpas
CPT4	Vincent Vidal
Fast Downward Autotune	Chris Fawcett, Malte Helmert, Holger Hoos, Erez Karpas, Gabi Röger, Jendrik Seipp
Fast Downward Stone Soup-1	Malte Helmert, Jörg Hoffmann, Erez Karpas, Emil Keyder, Raz Nissim, Silvia Richter, Gabi Röger, Jendrik Seipp, Matthias Westphal
Fast Downward Stone Soup-2	Malte Helmert, Jörg Hoffmann, Erez Karpas, Emil Keyder, Raz Nissim, Silvia Richter, Gabi Röger, Jendrik Seipp, Matthias Westphal
Fork Init	Michael Katz, Carmel Domshlak
Gamer	Peter Kissmann, Stefan Edelkamp
IFork Init	Michael Katz, Carmel Domshlak
LM-cut	Malte Helmert, Carmel Domshlak
LMFork	Michael Katz, Carmel Domshlak
Merge and Shrink	Raz Nissim, Malte Helmert, Jörg Hoffmann
Selective Max	Erez Karpas, Carmel Domshlak, Malte Helmert, Shaul Markovitch

Tabla 3 – *Listado de planificadores: Óptima Secuencial*

2.7.2.2 *Resultados obtenidos*

Como en el apartado anterior, se muestra una figura resumen de los resultados obtenidos por los planificadores (filas) en cada uno de los dominios (columnas), con un código de color en gama de grises a modo de indicativo del volumen de problemas resueltos.

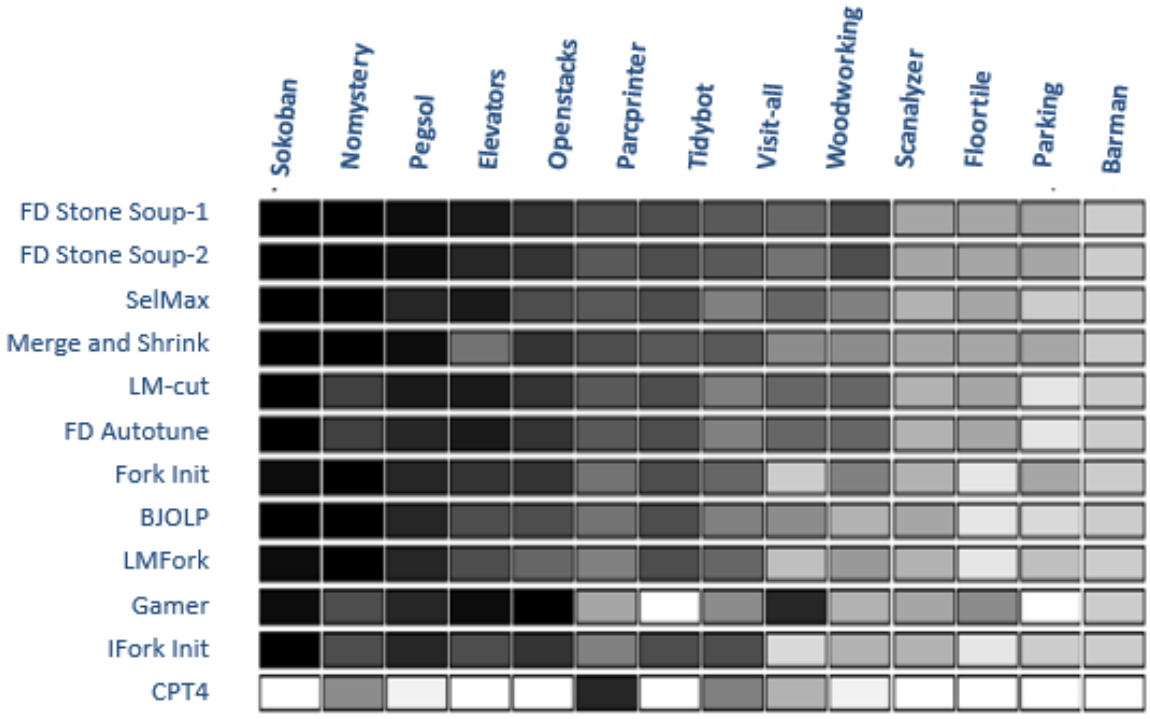

Figura 5 - Resultados categoría óptima secuencial

Las dos versiones de Fast Downward Stone Soup fueron las ganadoras seguidas de los subcampeones ex–aequo Selective Max y Merge and Shrink.

2.7.2.3 *Planificadores a destacar*

2.7.2.3.1 *SelMax*

El planificador Selective Max, SelMax, en un alto nivel, podríamos decir que SelMax es un planificador de hyperheurísticas, una heurística para elegir entre otras heurísticas.

2.7.2.3.2 Merge & Shrink

El planificador Merge and Shrink, M&S, utiliza la abstracción como enfoque general para el diseño heurístico. Es decir, utiliza soluciones calculadas en un espacio de estados abstractos más pequeño para ofrecer una función heurística consistente y admisible, permitiendo seleccionar pares individuales de los estados a agregar.

El planificador está implementado partiendo del sistema de planificación Fast Downward.

2.7.3 Planificadores de la categoría Multi-core

Recordar que, el objetivo de los planificadores de esta categoría de la competición, es minimizar el coste de las acciones (suma de *action-costs);* basándose en STRIPS + *action-costs* como en partes anteriores, pero en esta ocasión a los planificadores les está permitido el uso de todas las *cores* o núcleos simultáneamente.

2.7.3.1 Listado de planificadores

A continuación, se muestra el listado de los 7 planificadores participantes junto con los autores de los mismos.

PLANIFICADOR	AUTORES
ACOPlan	Marco Baioletti, Alfredo Milani, Valentina Poggioni, Fabio Rossi
Arvand Herd	Hootan Nakhost, Martin Mueller, Jonathan Schaeffer, Nathan Sturtevant, Richard Valenzano
ay-Also-Plan Threaded	Juhan Ernits, Charles Gretton
Roamer-p	You Xu, Qiang Lu, Ruoyun Huang, Yixin Chen
Madagascar	Jussi Rintanen
Madagascar-p	Jussi Rintanen
PHSFF	Moisés Martínez

Tabla 4 - *Listado de planificadores: Mult-icore*

2.7.3.2 Resultados obtenidos

A continuación, se muestra una figura que resume los resultados obtenidos por los planificadores en cada uno de los dominios. El código de color en gama de grises es indicativo de la cantidad de problemas resueltos por cada planificador.

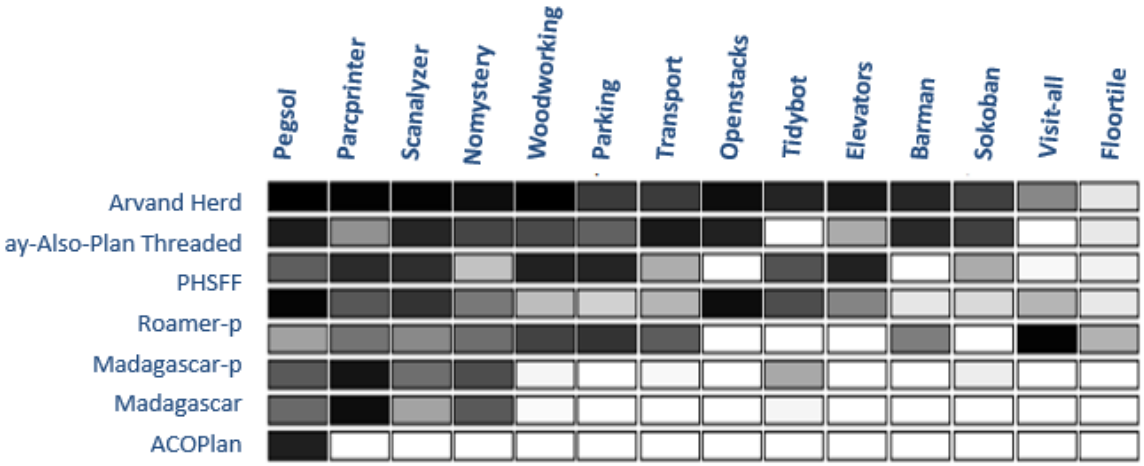

Figura 6 – Resultados categoría Multi-Core

El planificador con mayor puntuación de esta fase de la competición fue Arvand Herd, seguido por ay-Also-Plan Threaded.

2.7.3.3 Planificadores a destacar

2.7.3.3.1 Arvand Herd

Se ha demostrado que, para lograr el mejor rendimiento posible, dado el conjunto diverso de dominios, a menudo es necesario utilizar diferentes configuraciones de un algoritmo para cada uno de los diferentes problemas.

ArvandHerd es un planificador en paralelo introducido en la IPC de 2011, que utiliza un enfoque basado en un portfolio de configuraciones con cuatro parametrizaciones distintas del planificador Arvand y una del planificador LAMA. Cada procesador ejecuta un solo planificador independientemente de los otros a fin de minimizar la sobrecarga debido a la comunicación.

La arquitectura general de ArvandHerd incluye la comunicación entre los procesadores, gestión de memoria y el uso del sistema de mejora del plan Aras, para mejorar la calidad del plan obtenido.

2.7.3.3.2 ay-Also-Plan Threaded

Este planificador, AYALSOPLAN, hace su entrada en la categoría Multi-Core de la IPC 2011 intentando dar respuesta a los problemas de memoria con un enfoque que trata de paralelizar el procesamiento para aprovechar la proliferación de *clusters* y equipos modernos con varios procesadores, para no agotar la memoria del ordenador con una sobrecarga de almacenamiento.

A tener en cuenta que, el procesamiento en paralelo cesa si un ordenador de un solo núcleo puede construir de manera eficiente el plan.

2.7.4 Planificadores de la categoría Temporal

Basada en STRIPS + *action costs* (+ *fluents* numéricos) con el objetivo de que los planificadores minimicen el tiempo total, *makespan*, del plan.

2.7.4.1 Listado de planificadores

A continuación, se muestra el listado de los 8 planificadores participantes junto con los autores de los mismos.

PLANIFICADOR	AUTORES
CPT4	Vincent Vidal
DAEYAHSP	Johann Dréo, Marc Schoenauer, Pierre Savéant, Vincent Vidal
LMTD	Yanmei Hu, Dunbo Cai, Minghao Yin
POPF2	Amanda Coles, Andrew Coles, Maria Fox, Derek Long
Sharaabi	Bharat Ranjan Kavuluri
TLP-GP	Frederic Maris, Pierre Regnier
YAHSP2	Vincent Vidal
YAHSP2-MT	Vincent Vidal

***Tabla 5** - Listado de planificadores: Temporal*

2.7.4.2 Resultados obtenidos

Como en apartados anteriores, mostramos una figura resumen con los resultados obtenidos por los planificadores en cada uno de los dominios. El código de color en gama de grises indica grosso modo el volumen de problemas resueltos por cada planificador.

Los planificadores premiados fueron en primer lugar DAEYAHSP, seguido por YAHSP2-MT y POPF2 con el mismo número de puntos.

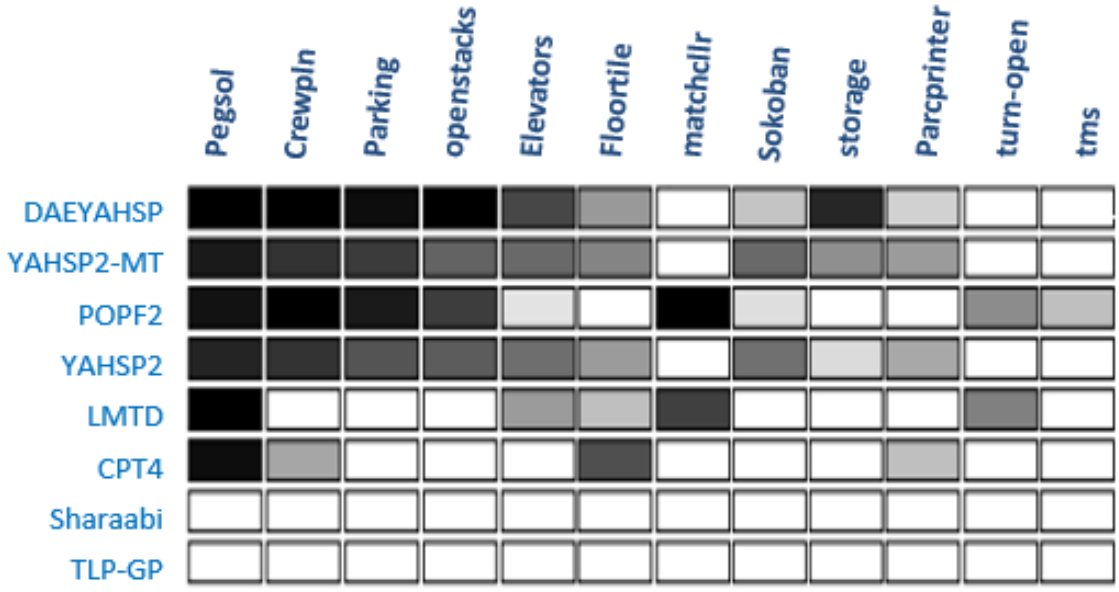

Figura 7 – Resultados categoría Temporal

2.7.4.3 Planificadores a destacar

2.7.4.3.1 YAHSP2-MT

Su diseño ha tenido por objetivo la simplicidad, tanto en los algoritmos como en el código fuente con lo que ha sufrido muchas modificaciones con respecto a las primeras versiones.

YAHSP2 se basa en la computación de planes con estrategia de búsqueda hacia delante a partir de planes relajados a través de la búsqueda heurística en el espacio de estados.

Fue el primer planificador en publicar esta idea en el año 2003 demostrando que esta simple idea sigue siendo vigente, haciendo a los planificadores muy eficientes en comparación con el estado del arte, en términos de tiempo de ejecución y número acumulado de problemas resueltos.

Como podemos comprobar, no trata un tema tan importante como es la calidad del plan, ya que el objetivo es producir un planificador rápido y fácilmente integrable en un sistema más amplio que sí tenga en cuenta esta calidad, como puede ser el planificador DAEx.

2.7.4.3.2 DAE YAHSP

DAEx, es la aplicación concreta del paradigma *Divide-and-Evolve*, divide y vencerás, impulsada por un algoritmo evolutivo. Es un sistema de planificación satisficing independiente del dominio basado en Computación Evolutiva.

Utiliza el principio de descomposición, un algoritmo evolutivo para conducir el proceso de optimización, y un planificador X integrado para resolver los sub-problemas.

La configuración presentada en la competición es DAE YAHSP. Una conjunción de instancias de DAEx, con heurísticas de búsqueda hacia delante del planificador YAHSP. En este matrimonio, YAHSP realiza varios intentos para encontrar una solución rápidamente mientras que DAE controla el proceso de optimización.

2.7.4.3.3 POPF2

El objetivo de POPF2 es mantener los beneficios de la construcción de un plan de orden parcial, es decir, de producir *makespan* eficiente, planes flexibles y evitar la aparición de conflictos ampliando el plan siempre en la misma dirección. Este planificador trabaja bajo la semántica de PDDL 2.1.

3. Análisis

3.1 Selección del dominio de pruebas

Para seleccionar el dominio de pruebas, se han tenido en cuenta aquellos dominios que tengan más posibilidades de modelarse de formas alternativas a la usada en la competición, utilizando fragmentos de PDDL no contemplados en ella. Por este motivo, era conveniente seleccionar alguno que incluyera *:action-costs*.

El siguiente cuadro, muestra los Requerimientos incluidos originalmente en cada dominio:

	:requirements			
DOMINIOS	:strips	:typing	:action-costs	:equality
barman	x	x	x	
elevators		x	x	
floortile		x		
nomystery		x	x	
openstacks		x	x	
parcprinter		x	x	
parking	x	x	x	
pegsol		x	x	
scanalyzer		x	x	
sokoban		x	x	
tidybot	x	x		x
transport		x	x	
visitall		x		
woodworking		x	x	

***Tabla 6** –Requerimientos por dominio*

Una vez evaluados tanto dominio como requerimientos, se ha seleccionado el dominio '*elevators*' para la realización de las modificaciones de modelado puesto que el problema tiene amplias posibilidades de modelado en cuanto al intercambio de precondiciones por funciones, disminuyendo el volumen del código considerablemente (como puede verse en el apartado de anexos).

3.2 Selección de los planificadores

Inicialmente se ha considerado trabajar con 4 planificadores diferentes:

· cbp2:

La ventaja de trabajar con este planificador es que, se supone, es capaz de minimizar el coste de los planes y no solamente el número de pasos. Además, acepta precondiciones numéricas.

· ibacop2:

El ganador de la última competición IPC 2014. Es un portfolio de distintos planificadores cada uno corriendo un determinado tiempo. No acepta precondiciones numéricas. Por este motivo, con este planificador comprobaremos la calidad del plan sin modificaciones en el dominio, comparándolo con los resultados obtenidos, tras el modelado, por el resto de planificadores.

· madagascar:

Con un paradigma de planificación totalmente distinto al resto (planificación SAT). No soporta precondiciones numéricas.

· Popf2:

Soporta precondiciones numéricas, aunque no precondiciones negativas. Una buena opción para comparar con los resultados de cbp2.

3.3 Detalle de las Modificaciones

En este apartado, se pasan a detallar brevemente las modificaciones realizadas sobre el dominio seleccionado, ***Elevators***, y sus problemas asociados, así como la batería de pruebas que se va a realizar con dichas modificaciones, para cada planificador.

El objetivo es simplificar el dominio (y los problemas del mismo), de modo que podamos comprobar si los planificadores encuentran un plan mejor o no, demostrando la importancia que tiene el modelado en la planificación.

· Modificación 1.- Se elimina el predicado ***can-hold***, sustituyéndolo por la función ***capacity.*** Con ello se redefine la capacidad del elevador, asignándole un valor numérico.

```
(:functions
        (capacity  ?lift - elevator) - number
)
```

· Modificación 2.- Se eliminan los predicados ***passengers*** *y* ***next***, sustituyéndolos por la función ***occupation.*** Complementa a la función anterior ya que, de no hacer uso de la función anterior no se podría eliminar el predicado *next.* También se eliminan objetos de tipo – *count* de los parámetros de las acciones, simplificando aún más el dominio.

```
(:functions
        (occupation  ?lift - elevator) - number
)
```

· Modificación 3.- Se elimina el predicado **above** y se sustituye por la función ***floor-number.***

```
(:functions
        (floor-number ?floor - count) - number
)
```

4. Pruebas

4.1 Requisitos de pruebas

Para llevar a cabo las pruebas de los dominios ha sido necesario tener en cuenta unos requisitos de tiempo y memoria para que todas las pruebas se realicen en las mismas condiciones.

Los problemas se lanzarán con un script de manera secuencial limitando, en este caso, el tiempo que tiene cada planificador para resolver cada problema a 600 segundos (10 minutos).

Del mismo modo, como las pruebas se han realizado en un ordenador con 3 GB de memoria RAM, se ha restringido la memoria que puede usar cada planificador en cada problema a 2'8 GB. Todas las pruebas se han realizado en un ordenador Intel(R) Xeon(TM) 3GHz con sistema operativo Ubuntu 10.04.4 LTS.

4.2 Casos de Prueba

Para obtener los casos de prueba se ha ido modelando el dominio de manera que, a través de los diferentes casos, se prueben las diferentes modificaciones individual y conjuntamente, para luego compararlas con los resultados obtenidos al probar el dominio original, sin ninguna modificación.

Teniendo en cuenta las modificaciones descritas anteriormente, se ha elaborado el siguiente cuadro en el cual se muestra la batería de pruebas con las que se ha trabajado, indicando para cada prueba, qué modificación o modificaciones tiene incluidas:

MODIFICACIONES SOBRE EL DOMINIO			
	Modif.3	Modif.2	Modif.1
Prueba1	0	0	0
~~Prueba2*~~	~~0~~	~~0~~	~~1~~
~~Prueba3**~~	~~0~~	~~1~~	~~0~~
Prueba4	0	**1**	**1**
Prueba5	**1**	0	0
~~Prueba6*~~	~~1~~	~~0~~	~~1~~
~~Prueba7*~~	~~**1**~~	~~**1**~~	~~0~~
Prueba8	**1**	**1**	**1**

Tabla 7 *– Batería de pruebas dominio Elevators*

* Prueba2 y Prueba6: se eliminan las pruebas de estas modificaciones ante la imposibilidad de evaluar la capacidad de un ascensor sin conocer su ocupación.

** Pruebas3 y Prueba7: de igual manera que las pruebas Prueba2 y Prueba6, no pueden realizarse las pruebas de las modificaciones 1 y 2 independientemente. Además, eso nos costaría seguir manteniendo el predicado *next*.

5. Resultados

5.1 Resultados obtenidos

A continuación, se muestran los cuadros de resultados de cada planificador, para cada uno de los 20 problemas de los que se componía la batería de problemas de este dominio en la competición IPC-2011.

Los resultados se presentan en forma de coste del plan obtenido según el dominio probado. Si el planificador, en el tiempo que tiene asignado para ello, ha encontrado más de un plan válido para el mismo problema, los costes de esos planes-por-problema se han incluido en los cuadros separados por ';'.

5.1.1 Resultados cbp2

Al haber restringido el tiempo de pruebas a 10 minutos, el planificador no es capaz de encontrar solución a todos los problemas.

En cualquier caso y como se muestra en el siguiente cuadro, con las modificaciones de los dominios aumenta el número de problemas a los que el planificador encuentra solución. Y aunque, el número de soluciones por problema no se vea afectado significativamente en la mayoría de los casos, si es cierto que el coste total de los planes es menor.

	PLANIFICADOR CBP2: COSTES POR DOMINIO			
PROBLEMAS	Dominio Sin modificaciones	Dominio con Modif.1+Modif.2+Modif.3	Dominio con Modif.1+Modif.2	Dominio con Modif.3
P01	404; 352; 327; 314; 280	282; 281; 277; 271; 266 261; 257; 252; 242	282; 281; 277; 271; 266 261; 257; 252; 242	404; 352; 327; 314; 280
P02	1242;1037; 959; 853; 848	632; 604; 592; 587; 555; 532; 436; 417	632; 604; 592; 587; 555; 532; 436; 417; **396**	1242;1037; 959; 853; 848
P03	**1368; 1237; 1213; 1193; 1103**	780; 758	780; 758	-
P04	714; 704; 644; 640	519; 400; 387	519; 400; 387	714; 704; 644; 640
P05	-	300; 299; 291	300; 299; 291	-
P06	1279; 1255; 1112	653	653	1279; 1255; 1112
P07	979; 969; 944; 897	629; 604: 545; 484	629; 604: 545; 484	979; 969; 944; 897
P08	1406; 1349; 1215; 1174	984; 822; 820; 807	984; 822; 820; 807	1406; 1349; 1215; 1174
P09	1745; 1524; 1386	857; 768; 675	857; 768; 675	1745; 1524; 1386
P10	-	1243	1243	-
P11	1885; 1866	-	**1258; 1058; 965; 961**	1885; 1866
P12	-	1327; 994; 891; 800	1327; 994; 891; 800	-
P13	-	1273; 1249; 1172; 1027	1273; 1249; 1172; 1027	-
P14	-	2292; 2273	2292; 2273	-
P15	2624	1472; 1190; 1084;1059; 1040	1472; 1190; 1084;1059; 1040	2624
P16	-	-	-	-
P17	-	-	-	-
P18	-	-	-	-
P19	-	-	-	-
P20	-	-	-	-

***Tabla 8** – Cuadro de resultados cbp2*

En el cuadro anterior, podemos ver marcado en negrita los resultados con diferencias a destacar entre las pruebas realizadas en las modificaciones del dominio. Los resultados obtenidos en las pruebas con los dominios que incluyen las modificaciones 'Modif.1+Modif.2+Modif.3' y 'Modif.1+Modif.2' obtienen similar resultado excepto para los problemas p02, donde la prueba con el menor modelado ha obtenido mejor resultado y p11, donde no ha encontrado solución. Del mismo modo las pruebas con el dominio original sin modificar y el dominio que incluyen la modificación 'Modif.3', también obtienen resultados parecidos salvo en el problema marcado en negrita (p03).

A continuación, podemos ver el cuadro simplificado con el plan de menor coste encontrado por cada planificador en cada problema.

	PLANIFICADOR CBP2: COSTES POR DOMINIO			
PROBLEMAS	Dominio Sin modificaciones	Dominio con Modif.1+Modif.2+Modif.3	Dominio con Modif.1+Modif.2	Dominio con Modif.3
P01	280	242	242	280
P02	848	417	396	848
P03	1103	758	758	-
P04	640	387	387	640
P05	-	291	291	-
P06	1112	653	653	1112
P07	897	484	484	897
P08	1174	807	807	1174
P09	1386	675	675	1386
P10	-	1243	1243	-
P11	1866	-	961	1866
P12	-	800	800	-
P13	-	1027	1027	-
P14	-	2273	2273	-
P15	2624	1040	1040	2624
P16	-	-	-	-
P17	-	-	-	-
P18	-	-	-	-
P19	-	-	-	-
P20	-	-	-	-

Tabla 9 *– Cuadro de resultados simplificado cbp2*

De estas ejecuciones podemos intuir que la modificación Modif.3 por sí sola o en combinación con las otras dos modificaciones no aporta mejoras en cuestión de costes del plan o problemas resueltos, comparándolo con el dominio sin modificaciones.

Se ha podido comprobar que, inicialmente, aporta mejoras en cuanto al tiempo en que tarda el planificador en encontrar planes válidos, pero las pruebas realizadas sin la modificación Modif.3, han encontrado mayor número de soluciones a los problemas con lo que, podríamos decir, esta modificación ha sido contraproducente.

Del mismo modo las modificaciones 'Modif.1+Modif.2+Modif.3' y 'Modif.1+Modif.2' no distan mucho entre ellas (es decir, que la inclusión de la Modif.3 no aporta mejoras). Pero ambas, respecto al dominio original, suponen enormes mejoras de coste en los planes.

5.1.2 Resultados ibacop

Este planificador, al no soportar precondiciones numéricas, como comentamos anteriormente nos servirá como punto de referencia con los otros planificadores.

Se han realizado únicamente pruebas con él, para el dominio original sin modificar. Al igual que sucede con el planificador cbp2, al restringir el tiempo de búsqueda de planes por problema a 10 min., el planificador no es capaz de encontrar solución a todos los problemas.

	PLANIF. IBACOP: COSTES POR DOMINIO	
PROBLEMAS	Dominio Sin modificaciones	Dominios Modificados
P01	525; 398; 303; 235; 207; 192	-
P02	977; 842; 366	-
P03	826; 430	-
P04	616; 268	-
P05	444; 363	-
P06	983; 524	-
P07	773; 452	-
P08	994; 510	-
P09	1327	-
P10	-	-
P11	-	-
P12	-	-
P13	-	-
P14	-	-
P15	-	-
P16	-	-
P17	-	-
P18	-	-
P19	-	-
P20	-	-

***Tabla 10** – Cuadro de resultados ibacop*

La misma tabla anterior simplificada, mostrando el plan de menor coste encontrado por el planificador para cada problema.

	PLANIF. IBACOP: COSTES POR DOMINIO	
PROBLEMAS	Dominio Sin modificaciones	Dominios Modificados
P01	192	-
P02	366	-
P03	430	-
P04	268	-
P05	363	-
P06	524	-
P07	452	-
P08	510	-
P09	1327	-
P10	-	-
P11	-	-
P12	-	-
P13	-	-
P14	-	-
P15	-	-
P16	-	-
P17	-	-
P18	-	-
P19	-	-
P20	-	-

Tabla 11 *– Cuadro de resultados simplificado ibacop*

5.1.3 Resultados Madagascar

Para este dominio, el planificador Madagascar no es capaz de encontrar en el tiempo que hemos definido ninguna solución.

5.1.4 Resultados Popf2

Como en planificadores anteriores y por los motivos ya citados, no se han encontrado planes válidos para todos los problemas.

De igual manera que con el planificador cbp2, con las modificaciones de los dominios hemos aumentado el número de problemas a los que el planificador encuentra solución.

En cuanto a la calidad de los planes entendida en coste, aunque podemos intuir una mejora en la mayoría de las pruebas, no podemos asegurar con esta batería de pruebas que el modelado del dominio mejore en todos los casos la misma; según se aprecia en el siguiente cuadro con datos marcados en rojo.

	PLANIFICADOR POPF2: COSTES POR DOMINIO			
PROBLEMAS	Dominio Sin modificaciones	Dominio con Modif.1+Modif.2+Modif.3	Dominio con Modif.1+Modif.2	Dominio con Modif.3
P01	338; 284; 281	374; 238; 233	374; 238; 233	338; 284; 281
P02	1234	1058	1058	1234
P03	**1115**	1057; 1027	1057; 1027	-
P04	594; 466; 463	459; 388	459; 388	594; 466; 463
P05	505; 456; 450	463; 454; 448	463; 454; 448	505; 456; 450
P06	1292	996	996	1292
P07	1111	1159; 987; 986	1159; 987; 986	1111
P08	-	1768	1768	-
P09	-	1340	1340	-
P10	-	1173	1173	-
P11	-	-	**1478**	-
P12	-	-	**1515**	-
P13	-	-	-	-
P14	-	-	-	-
P15	-	-	-	-
P16	-	-	-	-
P17	-	-	-	-
P18	-	-	-	-
P19	-	-	-	-
P20	-	-	-	-

***Tabla 12**– Cuadro de resultados popf2*

En negrita, en el cuadro anterior, se han marcado los resultados a destacar. Las pruebas realizadas incluyendo la modificación Modif.3 han provocado que se encuentre menor número de soluciones a los problemas (problemas p03, p11 y p12). Esta modificación no ha supuesto mejora.

A continuación, podemos ver el cuadro simplificado con el plan de menor coste encontrado por el planificador en cada problema.

	PLANIFICADOR POPF2: COSTES POR DOMINIO			
PROBLEMAS	Dominio Sin modificaciones	Dominio con Modif.1+Modif.2+Modif.3	Dominio con Modif.1+Modif.2	Dominio con Modif.3
P01	281	233	233	281
P02	1234	1058	1058	1234
P03	1115	1027	1027	-
P04	463	388	388	463
P05	450	448	448	450
P06	1292	996	996	1292
P07	1111	986	986	1111
P08	-	1768	1768	-
P09	-	1340	1340	-
P10	-	1173	1173	-
P11	-	-	1478	-
P12	-	-	1515	-
P13	-	-	-	-
P14	-	-	-	-
P15	-	-	-	-
P16	-	-	-	-
P17	-	-	-	-
P18	-	-	-	-
P19	-	-	-	-
P20	-	-	-	-

Tabla 13 *- Cuadro de resultados simplificado popf2*

5.2 Cálculo de calidad y ratios de costes

	PLANIFICADORES: COSTES POR DOMINIO									
	SIN MODIFICACIONES				Modif.1+Modif.2+Modif.3		Modif.1+Modif.2		Modif.3	
PROBLEMAS	PLANIF. CBP2	PLANIF. POPF2	PLANIF. IBACOP	PLANIF. MADAGASCAR	PLANIF. CBP2	PLANIF. POPF2	PLANIF. CBP2	PLANIF. POPF2	PLANIF. CBP2	PLANIF. POPF2
P01	280	281	192	-	242	233	242	233	280	281
P02	848	1234	366	-	417	1058	396	1058	848	1234
P03	1103	1115	430	-	758	1027	758	1027	-	-
P04	640	463	268	-	387	388	387	388	640	463
P05	-	450	363	-	291	448	291	448	-	450
P06	1112	1292	524	-	653	996	653	996	1112	1292
P07	897	1111	452	-	484	986	484	986	897	1111
P08	1174	-	510	-	807	1768	807	1768	1174	-
P09	1386	-	1327	-	675	1340	675	1340	1386	-
P10	-	-	-	-	1243	1173	1243	1173	-	-
P11	1866	-	-	-	-	-	961	1478	1866	-
P12	-	-	-	-	800	-	800	1515	-	-
P13	-	-	-	-	1027	-	1027	-	-	-
P14	-	-	-	-	2273	-	2273	-	-	-
P15	2624	-	-	-	1040	-	1040	-	2624	-
P16	-	-	-	-	-	-	-	-	-	-
P17	-	-	-	-	-	-	-	-	-	-
P18	-	-	-	-	-	-	-	-	-	-
P19	-	-	-	-	-	-	-	-	-	-
P20	-	-	-	-	-	-	-	-	-	-

***Tabla 14**– Cuadro resumen de resultados*

A partir de los cuadros de resultado simplificados por planificador, hemos obtenido la anterior tabla resumen con los menores costes de cada plan para cada problema, según el planificador y el dominio ejecutado.

Como puede comprobarse, ibacop es el planificador que genera los planes con mejores resultados, seguido por cbp2 cuyos resultados en las pruebas con el dominio modificado se aproximan más a los resultados de ibacop que a los resultados que se obtienen con el dominio sin modelar.

El modelado del dominio ha producido también un aumento claro del número de problemas a los que los planificadores encuentran solución, superando en ambos casos (planificador cbp2 y popf2 sobre dominio modificado) al número de soluciones encontradas por ibacop.

Seleccionando únicamente las pruebas con el modelado que ha obtenido mejores resultados (aquellas que incluyen las modificaciones 'Modif.1+Modif.2'), vamos a compararlas con las pruebas del dominio sin modificar. Los siguientes ratios de coste nos indican la calidad de los planes de cada planificador para que podamos compararlos entre ellos de manera más clara.

Al igual que en la competición IPC, vamos a asignarle a cada problema una puntuación entre 0 y 1, resultado de dividir el coste mínimo obtenido por cualquier planificador, por el coste obtenido por cada planificador en cada problema. De este modo, el planificador que obtenga más puntuación global es el mejor.

	DOMINIO SIN MODIFICACIONES			**DOMINIO MODIFICADO (Modif.1+Modif.2)**	
PROBLEMAS	PLANIF. CBP2	PLANIF. POPF2	PLANIF. IBACOP	PLANIF. CBP2	PLANIF. POPF2
P01	0,69	0,68	1	0,79	0,82
P02	0,43	0,30	1	0,92	0,35
P03	0,39	0,39	1	0,57	0,42
P04	0,42	0,58	1	0,69	0,69
P05	-	0,65	0,80	1	0,65
P06	0,47	0,41	1	0,80	0,53
P07	0,50	0,41	1	0,93	0,46
P08	0,43	-	1	0,63	0,29
P09	0,49	-	0,51	1	0,50
P10	-	-	-	0,94	1,00
P11	0,52	-	-	1	0,65
P12	-	-	-	1	0,53
P13	-	-	-	1	-
P14	-	-	-	1	-
P15	0,40	-	-	1	-
PUNTUACIÓN	4,73	3,40	8,31	13,29	6,88

Tabla 15 *– Ratios de costes por planificador*

Como comentábamos anteriormente, aunque ibacop es el planificador que obtiene los planes de mejor calidad, con una puntuación global en pruebas (en la tabla anterior) de 8,31 puntos, las pruebas con el dominio modificado han provocado que se vea superado por cbp2, con puntuación global en pruebas de 13,29 puntos, al encontrar este un mayor número de resultados a los problemas.

Del cuadro anterior podemos deducir la mejora que han sufrido los planificadores sometidos a las pruebas de modelado:

	PLANIF. CBP2			**PLANIF. POPF2**		
PROBLEMAS	DOM. SIN MODIF.	DOM. MODIFIC.	MEJORA	DOM. SIN MODIF.	DOM. MODIFIC.	MEJORA
P01	0,69	0,79	10,77%	0,68	0,82	14,07%
P02	0,43	0,92	49,26%	0,30	0,35	4,93%
P03	0,39	0,57	17,75%	0,39	0,42	3,30%
P04	0,42	0,69	27,37%	0,58	0,69	11,19%
P05	-	1	100,00%	0,65	0,65	0,29%
P06	0,47	0,80	33,13%	0,41	0,53	12,05%
P07	0,50	0,93	43,00%	0,41	0,46	5,16%
P08	0,43	0,63	19,76%	0,68	0,29	28,85%
P09	0,49	1	51,30%	0,30	0,50	50,37%
P10	-	0,94	94,37%	0,39	1,00	100,00%
P11	0,52	1	48,50%	0,58	0,65	65,02%
P12	-	1	100,00%	0,65	0,53	52,81%
P13	-	1	100,00%	0,41	-	-
P14	-	1	100,00%	0,41	-	-
P15	0,40	1	60,37%	0,68	-	-

Tabla 16 *– Mejoras del modelado por planificador*

La mejora total es de 8,56 puntos en la puntuación global para cbp2 y de 3,48 puntos para popf2. Esto hace una mejora en la puntuación de más del doble de la obtenida en las pruebas con dominio sin modelar.

5.3 Evaluación de resultados.

Con el fin de clarificar las conclusiones, vamos a descartar las pruebas de los dominios con modificaciones 'Modif.3' y 'Modif.1+Modif.2+Modif.3', entendiendo que, en este caso, una pequeña modificación del dominio (como es Modif.3) no ha supuesto mejora en el coste resultante del plan, mientras que un amplio modelado si nos permite distinguir mejoras considerables.

Tal como comentamos en el apartado anterior, el modelado del dominio ha supuesto una mejora en ambos planificadores, aunque en cbp2 ha sido más apreciable que en popf2. La calidad de los planes ha mejorado considerablemente acercándose a la del planificador ibacop en cuestión de costes y superándolo en número de problemas resueltos, entendiendo una mejora en el tiempo de resolución de los planificadores, dado que, en el mismo tiempo, consiguen resolver mayor número de problemas.

6. Conclusiones y Líneas Futuras

6.1 Conclusiones

En el presente apartado y una vez completada la realización del proyecto, queda analizar si se han cumplido los objetivos y metas inicialmente propuestas. Se han recogido las conclusiones obtenidas tanto a nivel personal como a nivel de proyecto, haciendo un mayor hincapié en aquellos puntos del desarrollo que han supuesto un mayor esfuerzo.

Recordando los subobjetivos marcados al inicio del proyecto y una vez finalizadas las fases de análisis (donde estudiamos y modificamos el dominio seleccionado), pruebas (donde realizamos las pruebas con los diferentes planificadores sobre los dominios y problemas con y sin modelar) y evaluación de los resultados, creo posible concluir que el modelado de los dominios juega un importante papel en la planificación automática, influyendo directa y significativamente en el número y calidad de los planes obtenidos por los planificadores. Con esto, se da por alcanzado con éxito el objetivo principal marcado al inicio del proyecto fin de carrera.

A nivel personal, es importante destacar los conocimientos adquiridos sobre planificación automática, comprensión y modificación del lenguaje PDDL y uso e interpretación de los planificadores, entre otros.

Ha sido todo un desafío la realización del proyecto y los conocimientos adquiridos me han servido como base para la solución de los problemas y necesidades del proyecto, además de suponer retos tanto por la parte técnica como por la parte profesional, que en un principio parecían imposibles de cumplir, pero que finalmente se han alcanzado con esfuerzo y dedicación.

6.2 Líneas Futuras

En pos de mejorar el proyecto fin de carrera una vez este ha finalizado, se presentan las distintas líneas futuras o líneas de futuras mejoras aplicables:

· Aumentar el número de dominios de planificación modelados, así como el número de casos de prueba. Como siguiente dominio a modelar se propone el dominio Transport en el que el uso de precondiciones numéricas también podría suponer una simplificación importante.

· Ampliar los requisitos de los planificadores en tiempos de ejecución y restricciones de memoria. Las pruebas de la IPC se realizan con 30 minutos y 6GB de memoria; para este PFC, al ser un estudio preliminar, se prefirió realizar pruebas más cortas para comprobar la validez de la aproximación.

· Extender las pruebas a otros planificadores.

7. Presupuesto

En este apartado se detalla el presupuesto estimado de este proyecto, desglosando los costes en función de su naturaleza, ya sean costes de personal, materiales o costes indirectos.

7.1 Desglose de tiempos por actividad

En la siguiente tabla se recoge el total de horas dedicadas a cada tarea de proyecto.

Tareas	Duración días	Duración horas
Propuesta	1	8
Estudio del entorno	25	200
Análisis de requisitos	10	80
Implementación	12	96
Pruebas	12	96
Documentación y cierre	20	160
Total		**640**

***Tabla 17** - Presupuesto. Resumen de tiempos por actividad*

En la elaboración del proyecto se han invertido **640 horas**, suponiendo que cada mes tiene 20 días laborables de 8 horas.

7.2 Costes de personal

La siguiente tabla muestra el coste total de cada uno de los diferentes roles a los que ha habido que adaptarse en cada una de las etapas y actividades que forman el proyecto.

Se ha tomado de referencia el coste persona/mes de una empresa de soluciones tecnológicas de desarrollo software para el sector informático. El cálculo del Coste-persona se realiza en función de la siguiente fórmula:

$$Coste\ personal = \frac{Total\ dias \times horas/dia}{Dedicacion\ de\ hombre/mes} = Coste\ hombre/mes$$

Dispondremos así, de un analista para realizar la tarea de análisis del entorno, un ingeniero senior que evaluará los requisitos y será el jefe del proyecto, un programador encargado de la implementación de las pruebas y dos ingenieros junior que realizarán las pruebas del sistema, análisis de resultados y documentación necesaria del proyecto.

Cargo	Horas dedicadas	Dedicación (Hombres/mes)*	Coste Hombres/mes (€)	Coste Total
Analista	208	1,58	2.000€	3.169,52€
Ingeniero Senior	80	0,61	3.000€	1.828,57€
Programador	96	0,73	1.800€	1.316,57€
Ingeniero Junior	96	0,73	1.200€	877,71€
Ingeniero Junior	160	1,22	1.200€	1.462,86€

Tabla 18 *– Presupuesto. Resumen costes de personal*

El coste total referente al personal del proyecto hace un total de **8.655,24€.**

*) 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas) Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

7.3 Costes materiales

Pasamos a detallar los gastos de los componentes hardware y software necesarios para el proyecto. El cálculo del coste imputable se realiza con la fórmula de cálculo de la Amortización, de la siguiente manera:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (100%)

Descripción	Coste	Dedicación (meses)	Coste Imputable
Ordenador para desarrollo	699€	4,14	48,23€
Ordenador para pruebas	650€	2,07	22,43€
Sistema Operativo Microsoft Windows 7 Professional	139€	4,88	11,31€
Sistema Operativo Linux (Ubuntu 10.04)	0€	3,66	0€
Openoffice.org	0€	4,88	0€
Herramientas de desarrollo y Pruebas (Planificadores)	0€	3,66	0€

Tabla 19 *– Presupuesto. Resumen costes materiales*

El coste total referente a los gastos materiales del proyecto es de **81,96€.**

7.4 Resumen del presupuesto

El coste total lo forma la suma de la amortización del coste material y el coste de la mano de obra. Para este proyecto es de **8.737,2€**. Los gastos indirectos, derivados de los posibles riesgos en su desarrollo, se consideran un 20% de los costes totales del proyecto, y ascenderían a **1.747,44€**.

Recurso	Coste Total
Coste de personal	8.655,24€
Costes materiales	81,96€
Costes indirectos	1.747,44€
Total sin I.V.A.	**10.484,64€**
Total con I.VA. (21%)	**12.686,41€**

***Tabla 20** – Resumen del presupuesto*

8. Anexos

A continuación, se inlcuye la definición completa de sintaxis BNF del lenguaje PDDL 3.1, traducida de la original publicada por Daniel L. Kovacs en 'Complete BNF description of PDDL 3.1 (completely corrected)' en 2011.

8.1 BNF definición del lenguaje PDDL 3.1

8.1.1 Descripción del Dominio

```
<domain>                        ::= (define (domain <nombre>)
                                        [<require-def>]
                                        [<types-def>]:typing
                                        [<constants-def>]
                                        [<predicates-def>]
                                        [<functions-def>]:fluents
                                        [<constraints>]
                                        <structure-def>*)
<require-def>                   ::= (:requirements <require-key>+)
<require-key>                   ::= Ver sección 1.3 Requerimientos
<types-def>                     ::= (:types <typed list (name)>)
<constants-def>                         ::= (:constants <typed list (nombre)>)
<predicates-def>                        ::= (:predicates <atomic formula skeleton>+)
<atomic formula skeleton>       ::= (<predicate> <typed list (variable)>)
<predicate>                     ::= <nombre>
<variable>                      ::= ?<nombre>
<atomic function skeleton>      ::= (<function-symbol> <typed list (variable)>)
<function-symbol>               ::= <nombre>
<functions-def>                         ::=:fluents (:functions <function typed list (atomic function
skeleton)>)
<function typed list (x)>       ::= x+ - <function type> <function typed list(x)>
<function typed list (x)>       ::=
<function typed list (x)>       ::=:numeric-fluents x+
                                Esto está en desuso desde PDDL 3.1, donde el tipo de 'fluent' por defecto es
                                numérico.
<function type>                         ::=:numeric-fluents number
<function type>                         ::=:typing + :object-fluents <type>
<constraints>                   ::=:constraints (:constraints <con-GD>)
<structure-def>                         ::= <action-def>
<structure-def>                         ::=:durative-actions <durative-action-def>
<structure-def>                         ::=:derived-predicates <derived-def>
<typed list (x)>                        ::= x*
<typed list (x)>                        ::=:typing x+ - <type> <typed list(x)>
<primitive-type>                        ::= <nombre>
<primitive-type>                        ::= object
<type>                                  ::= (either <primitive-type>+)
<type>                                  ::= <primitive-type>
<emptyOr (x)>                   ::= ()
<emptyOr (x)>                   ::= x
<action-def>                    ::= (:action <action-symbol>
                                        :parameters (<typed list (variable)>)
                                        <action-def body>)
<action-symbol>                         ::= <nombre>
<action-def body>               ::= [:precondition <emptyOr (pre-GD)>]
                                    [:effect <emptyOr (effect)>]
<pre-GD>                        ::= <pref-GD>
<pre-GD>                        ::= (and <pre-GD>*)
<pre-GD>                        ::=:universal-preconditions (forall (<typed list(variable)>) <pre-GD>)
<pref-GD>                       ::=:preferences (preference [<pref-name>] <GD>)
<pref-GD>                       ::= <GD>
<pref-name>                     ::= <nombre>
<GD>                            ::= <atomic formula(term)>
<GD>                            ::=:negative-preconditions <literal(term)>
<GD>                            ::= (and <GD>*)
<GD>                            ::=:disjunctive-preconditions (or <GD>*)
<GD>                            ::=:disjunctive-preconditions (not <GD>)
<GD>                            ::=:disjunctive-preconditions (imply <GD> <GD>)
<GD>                            ::=:existential-preconditions (exists (<typed list(variable)>) <GD> )
<GD>                            ::=:universal-preconditions (forall (<typed list(variable)>) <GD> )
<GD>                            ::=:numeric-fluents <f-comp>
<f-comp>                        ::= (<binary-comp> <f-exp> <f-exp>)
<literal(t)>                    ::= <atomic formula(t)>
<literal(t)>                    ::= (not <atomic formula(t)>)
<atomic formula(t)>             ::= (<predicate> t*)
```

```
<atomic formula(t)>          ::=:equality (= t t)
<term>                               ::= <nombre>
<term>                               ::= <variable>

<term>                               ::=:object-fluents <function-term>
<function-term>                      ::=:object-fluents (<function-symbol> <term>*)
<f-exp>                              ::=:numeric-fluents <number>
<f-exp>                              ::=:numeric-fluents (<binary-op> <f-exp> <f-exp>)
<f-exp>                              ::=:numeric-fluents (<multi-op> <f-exp> <f-exp>+)
<f-exp>                              ::=:numeric-fluents (- <f-exp>)
<f-exp>                              ::=:numeric-fluents <f-head>
<f-head>                             ::= (<function-symbol> <term>*)
<f-head>                             ::= <function-symbol>
<binary-op>                  ::= <multi-op>

<binary-op>                  ::= -
<binary-op>                  ::= /
<multi-op>                   ::= *
<multi-op>                   ::= +
<binary-comp>                ::= >
<binary-comp>                ::= <
<binary-comp>                ::= =
<binary-comp>                ::= >=
<binary-comp>                ::= <=
<nombre>                             ::= <letter> <any char>*
<letter>                             ::= a..z | A..Z
<any char>                   ::= <letter> | <digit> | - | _
<number>                             ::= <digit>+ [<decimal>]
<digit>                              ::= 0..9
<decimal>                    ::= .<digit>+
<effect>                             ::= (and <c-effect>*)
<effect>                             ::= <c-effect>
<c-effect>                   ::=:conditional-effects (forall (<typed list (variable)>) <effect>)
<c-effect>                   ::=:conditional-effects (when <GD> <cond-effect>)
<c-effect>                   ::= <p-effect>
<p-effect>                   ::= (not <atomic formula(term)>)
<p-effect>                   ::= <atomic formula(term)>
<p-effect>                   ::=:numeric-fluents (<assign-op> <f-head> <f-exp>)
<p-effect>                   ::=:object-fluents (assign <function-term> <term>)
<p-effect>                   ::=:object-fluents (assign <function-term> undefined)
<cond-effect>                ::= (and <p-effect>*)
<cond-effect>                ::= <p-effect>
<assign-op>                  ::= assign
<assign-op>                  ::= scale-up
<assign-op>                  ::= scale-down
<assign-op>                  ::= increase
<assign-op>                  ::= decrease
<durative-action-def>        ::= (:durative-action <da-symbol>
                                     :parameters (<typed list (variable)>)
                                     <da-def body>)
<da-symbol>                  ::= <nombre>
<da-def body>                ::= :duration <duration-constraint>
                                 :condition <emptyOr (da-GD)>
                                 :effect <emptyOr (da-effect)>

<da-GD>                              ::= <pref-timed-GD>
<da-GD>                              ::= (and <da-GD>*)
<da-GD>                              ::=:universal-preconditions (forall (<typed-list (variable)>) <da-GD>)
<pref-timed-GD>              ::= <timed-GD>
<pref-timed-GD>                      ::=:preferences (preference [<pref-name>] <timed-GD>)
<timed-GD>                   ::= (at <time-specifier> <GD>)
<timed-GD>                   ::= (over <interval> <GD>)
<time-specifier>                     ::= start
<time-specifier>                     ::= end
<interval>                   ::= all
<duration-constraint>        ::=:duration-inequalities (and <simple-duration-constraint>+)
<duration-constraint>        ::= ()
<duration-constraint>        ::= <simple-duration-constraint>
<simple-duration-constraint> ::= (<d-op> ?duration <d-value>)
<simple-duration-constraint> ::= (at <time-specifier> <simple-duration-constraint>)
<d-op>                               ::=:duration-inequalities <=
<d-op>                               ::=:duration-inequalities >=
<d-op>                               ::= =
<d-value>                    ::= <number>
<d-value>                    ::=:numeric-fluents <f-exp>
<da-effect>                  ::= (and <da-effect>*)
<da-effect>                  ::= <timed-effect>
<da-effect>                  ::=:conditional-effects (forall (<typed list (variable)>) <da-effect>)
<da-effect>                  ::=:conditional-effects (when <da-GD> <timed-effect>)
<timed-effect>               ::= (at <time-specifier> <cond-effect>)
<timed-effect>               ::=:numeric-fluents (at <time-specifier> <f-assign-da>)
<timed-effect>               ::=:continuous-effects + :numeric-fluents (<assign-op-t> <f-head> <f-exp-t>)
<f-assign-da>                ::= (<assign-op> <f-head> <f-exp-da>)
<f-exp-da>                   ::= (<binary-op> <f-exp-da> <f-exp-da>)
<f-exp-da>                   ::= (<multi-op> <f-exp-da> <f-exp-da>+)
<f-exp-da>                   ::= (- <f-exp-da>)
<f-exp-da>                   ::=:duration-inequalities ?duration
<f-exp-da>                   ::= <f-exp>
<assign-op-t>                ::= increase
<assign-op-t>                ::= decrease
<f-exp-t>                    ::= (* <f-exp> #t)
<f-exp-t>                    ::= (* #t <f-exp>)
<f-exp-t>                    ::= #t
<derived-def>                ::= (:derived <atomic formula skeleton> <GD>)
```

8.1.2. Descripción del Problema

```
<problem>                       ::= (define (problem <nombre>)
                                        (:domain <nombre>)
                                        [<require-def>]
                                        [<object declaration>]
                                        <init>
                                        <goal>
                                        [<constraints>]:constraints
                                        [<metric-spec>]:numeric-fluents
                                        [<length-spec>])
<object declaration>            ::= (:objects <typed list (nombre)>)
<init>                                  ::= (:init <init-el>*)
<init-el>                       ::= <literal(nombre)>
<init-el>                       ::=:timed-initial-literals (at <number> <literal(nombre)>)
<init-el>                       ::=:numeric-fluents (= <basic-function-term> <number>)
<init-el>                       ::=:object-fluents (= <basic-function-term> <nombre>)
<basic-function-term>           ::= <function-symbol>
<basic-function-term>           ::= (<function-symbol> <nombre>*)
<goal>                                  ::= (:goal <pre-GD>)
<constraints>                   ::=:constraints (:constraints <pref-con-GD>)
<pref-con-GD>                   ::= (and <pref-con-GD>*)
<pref-con-GD>                   ::=:universal-preconditions (forall (<typed list (variable)>) <pref-
con-GD>)
<pref-con-GD>                   ::=:preferences (preference [<pref-name>] <con-GD>)
<pref-con-GD>                   ::= <con-GD>
<con-GD>                        ::= (and <con-GD>*)
<con-GD>                        ::= (forall (<typed list (variable)>) <con-GD>)
<con-GD>                        ::= (at end <GD>)
<con-GD>                        ::= (always <GD>)
<con-GD>                        ::= (sometime <GD>)
<con-GD>                        ::= (within <number> <GD>)
<con-GD>                        ::= (at-most-once <GD>)
<con-GD>                        ::= (sometime-after <GD> <GD>)
<con-GD>                        ::= (sometime-before <GD> <GD>)
<con-GD>                        ::= (always-within <number> <GD> <GD>)
<con-GD>                        ::= (hold-during <number> <number> <GD>)
<con-GD>                        ::= (hold-after <number> <GD>)
<metric-spec>                   ::=:numeric-fluents (:metric <optimization> <metric-f-exp>)
<optimization>                          ::= minimize
<optimization>                          ::= maximize
<metric-f-exp>                          ::= (<binary-op> <metric-f-exp> <metric-f-exp>)
<metric-f-exp>                          ::= (<multi-op> <metric-f-exp> <metric-f-exp>+)
<metric-f-exp>                          ::= (- <metric-f-exp>)
<metric-f-exp>                          ::= <number>
<metric-f-exp>                          ::= (<function-symbol> <nombre>*)
<metric-f-exp>                          ::= <function-symbol>
<metric-f-exp>                          ::= total-time
<metric-f-exp>                          ::=:preferences (is-violated <pref-name>)
<length-spec>                   ::= (:length [(:serial <integer>)] [(:parallel <integer>)])
                                La longitud de las especificaciones está obsoleta desde PDDL 2.1.
```

8.1.2.1 Suspensión de limitaciones (para la declaración de restricciones)

Si queremos incrustar operadores modales uno dentro de otro, entonces debemos utilizar estas reglas en lugar de los que están en la sección 1.2, respectivamente.

```
<con-GD>                        ::= (always <con2-GD>)
<con-GD>                        ::= (sometime <con2-GD>)
<con-GD>                        ::= (within <number> <con2-GD>)
<con-GD>                        ::= (at-most-once <con2-GD>)
<con-GD>                        ::= (sometime-after <con2-GD> <con2-GD>)
<con-GD>                        ::= (sometime-before <con2-GD> <con2-GD>)
<con-GD>                        ::= (always-within <number> <con2-GD> <con2-GD>)
<con-GD>                        ::= (hold-during <number> <number> <con2-GD>)
<con-GD>                        ::= (hold-after <number> <con2-GD>)
<con2-GD>                       ::= <con-GD>
<con2-GD>                       ::= <GD>
```

8.1.3 Requerimientos

Aquí se muestra una tabla de todos los requisitos en PDDL 3.1. Algunos requisitos implican otros; algunos son abreviaturas para conjuntos comunes de requisitos. Si en un dominio no se establece ningún requisito, se asume la declaración de requisitos para :strips.

:strips	STRIPS-style básicos: adds y deletes
:typing	Permite type names en la declaración de variables
:negative-preconditions	Permite not en la descripción de objetivos
:disjunctive-preconditions	Permite or en la descripción de objetivos
:equality	Soporta = como predicado incorporado
:existential-preconditions	Permite exists en la descripción de objetivos
:universal-preconditions	Permite forall en la descripción de objetivos
:quantified-preconditions	= :existential-preconditions + :universal-preconditions
:conditional-effects	Permite when en efectos de las acciones
:fluents	= :numeric-fluents + :object-fluents
:numeric-fluents	Permite la definición de funciones numéricas y el uso de efectos utilizando operadores de asignación y las condiciones previas aritméticas.
:adl	= :strips + :typing + :negative-preconditions + :disjunctive-preconditions + :equality + :quantified-preconditions + :conditional-effects
:durative-actions	Permite acciones durativas. Nótese que esto no implica :numeric-fluents.
:duration-inequalities	Permite limitaciones de duración en acciones durativas utilizando las desigualdades.
:continuous-effects	Permite a las acciones durativas afectar a los fluents continuamente durante la duración de las acciones.
:derived-predicates	Permite predicados cuyo valor cierto es definido por una fórmula
:timed-initial-literals	Permite al estado inicial especificar literales que se convertirán en realidad en un momento determinado.Implica :durative-actions
:preferences	Permite el uso de preferencias en precondiciones de las acciones y objetivos.
:constraints	Permite el uso de campos de restricciones en los archivos de dominio y de problemas. Estos pueden contener operadores modales apoyando las limitaciones de trayectoria.
:action-costs	Si este requisito se incluye en la especificación PDDL, el uso de fluents numéricos se habilita ((similar al requisito :numeric-fluents). Sin embargo, fluents numéricos sólo se pueden utilizar de ciertas maneras muy limitadas: 1. Los fluents numéricos no se pueden utilizar en cualquier condición (precondiciones, metas, condiciones de efectos condicionales, etc.). 2. Un fluent numérico sólo se puede utilizar como objetivo de un efecto si es 0-aria y llamado total-cost. Si se utiliza tal efecto, entonces el fluent de coste total debe inicializarse a 0 en el estado inicial. 3. El único uso permitido de fluents numéricos en efectos está en los efectos de la forma (increase(total-cost) <numeric-term>), dónde el <numeric-term> es o bien una constantes numérica no negativa, o bien de la forma (<function-symbol> <term>*).(<term> está interpretado aquí como se muestra en la gramática PDDL, es decir, es un símbolo variable o una constante objeto. Notese que este <term> no puede ser <function-term>, incluso si se utiliza el requisito de objetos fluents.) 4. Los fluent numéricos no pueden ser inicializados con valores negativos. 5. Si el problema tiene especificada :metric, el objetivo debe ser (minimize(total-cost)), o si también se fijan los requerimientos :durative-action se puede minimizar una combinación lineal del total-cost y total-time, con coeficientes no negativos. Téngase en cuenta que una acción puede tener múltiples efectos que aumentan (coste-total), lo que es particularmente útil en el contexto de efectos condicionales. Téngase en cuenta también que estas restricciones implican que (coste-total) nunca disminuye durante la ejecución del plan, es decir, los costes de las acciones nunca con negativos.

A continuación, se presentan adjuntos, los documentos con las versiones del dominio modificadas para la realización de las pruebas, así como un ejemplo de problema de cada dominio, en concreto los problemas p01 y p20.

En el dominio y problemas originales se ha marcado en negrita el código a modificar. Del mismo modo, en el dominio y problemas que incluyen todas las modificaciones, se ha marcado en negrita el código modificado.

8.2 Elevators sin modificaciones

8.2.1 Dominio original

```
(define (domain elevators-sequencedstrips)
  (:requirements :typing :action-costs)
  (:types      elevator - object
               slow-elevator fast-elevator - elevator
               passenger - object
               count - object
         )

(:predicates
     (passenger-at ?person - passenger ?floor - count)
     (boarded ?person - passenger ?lift - elevator)
     (lift-at ?lift - elevator ?floor - count)
     (reachable-floor ?lift - elevator ?floor - count)
     (above ?floor1 - count ?floor2 - count)
     (passengers ?lift - elevator ?n - count)
     (can-hold ?lift - elevator ?n - count)
     (next ?n1 - count ?n2 - count)
)

(:functions (total-cost) - number
            (travel-slow ?f1 - count ?f2 - count) - number
            (travel-fast ?f1 - count ?f2 - count) - number
)

(:action move-up-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (above ?f1 ?f2 )
(reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f1 ?f2))))

(:action move-down-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
```

```
  :precondition (and (lift-at ?lift ?f1) (above ?f2 ?f1 )
(reachable-floor ?lift ?f2) )


  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f2 ?f1))))

(:action move-up-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (above ?f1 ?f2 )
(reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f1 ?f2))))

(:action move-down-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (above ?f2 ?f1 )
(reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f2 ?f1))))

(:action board
  :parameters (?p - passenger ?lift - elevator ?f - count
?n1 - count ?n2 - count)
  :precondition (and  (lift-at ?lift ?f) (passenger-at ?p
?f) (passengers ?lift ?n1) (next ?n1 ?n2) (can-hold ?lift
?n2) )
  :effect (and (not (passenger-at ?p ?f)) (boarded ?p ?lift)
(not (passengers ?lift ?n1)) (passengers ?lift ?n2) ))

(:action leave
  :parameters (?p - passenger ?lift - elevator ?f - count
?n1 - count ?n2 - count)
  :precondition (and  (lift-at ?lift ?f) (boarded ?p ?lift)
(passengers ?lift ?n1) (next ?n2 ?n1) )
  :effect (and (passenger-at ?p ?f) (not (boarded ?p ?lift))
(not (passengers ?lift ?n1)) (passengers ?lift ?n2) ))

)
```

8.2.2 Problema 1

```
(define (problem elevators-sequencedstrips-p16_14_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16  - count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13  - passenger
fast0 fast1  - fast-elevator
slow0-0 slow1-0 - slow-elevator
)

(:init
(next n0 n1) (next n1 n2) (next n2 n3) (next n3 n4) (next n4 n5) (next n5 n6) (next n6 n7) (next n7 n8) (next n8 n9) (next n9 n10) (next n10 n11) (next n11 n12) (next n12 n13) (next n13 n14) (next n14 n15) (next n15 n16)

(above n0 n1) (above n0 n2) (above n0 n3) (above n0 n4) (above n0 n5) (above n0 n6) (above n0 n7) (above n0 n8) (above n0 n9) (above n0 n10) (above n0 n11) (above n0 n12) (above n0 n13) (above n0 n14) (above n0 n15) (above n0 n16)
(above n1 n2) (above n1 n3) (above n1 n4) (above n1 n5) (above n1 n6) (above n1 n7) (above n1 n8) (above n1 n9) (above n1 n10) (above n1 n11) (above n1 n12) (above n1 n13) (above n1 n14) (above n1 n15) (above n1 n16)
(above n2 n3) (above n2 n4) (above n2 n5) (above n2 n6) (above n2 n7) (above n2 n8) (above n2 n9) (above n2 n10) (above n2 n11) (above n2 n12) (above n2 n13) (above n2 n14) (above n2 n15) (above n2 n16)
(above n3 n4) (above n3 n5) (above n3 n6) (above n3 n7) (above n3 n8) (above n3 n9) (above n3 n10) (above n3 n11) (above n3 n12) (above n3 n13) (above n3 n14) (above n3 n15) (above n3 n16)
(above n4 n5) (above n4 n6) (above n4 n7) (above n4 n8) (above n4 n9) (above n4 n10) (above n4 n11) (above n4 n12) (above n4 n13) (above n4 n14) (above n4 n15) (above n4 n16)
(above n5 n6) (above n5 n7) (above n5 n8) (above n5 n9) (above n5 n10) (above n5 n11) (above n5 n12) (above n5 n13) (above n5 n14) (above n5 n15) (above n5 n16)
(above n6 n7) (above n6 n8) (above n6 n9) (above n6 n10) (above n6 n11) (above n6 n12) (above n6 n13) (above n6 n14) (above n6 n15) (above n6 n16)
(above n7 n8) (above n7 n9) (above n7 n10) (above n7 n11) (above n7 n12) (above n7 n13) (above n7 n14) (above n7 n15) (above n7 n16)
(above n8 n9) (above n8 n10) (above n8 n11) (above n8 n12) (above n8 n13) (above n8 n14) (above n8 n15) (above n8 n16)
```

(above n9 n10) (above n9 n11) (above n9 n12) (above n9 n13) (above n9 n14) (above n9 n15) (above n9 n16)
(above n10 n11) (above n10 n12) (above n10 n13) (above n10 n14) (above n10 n15) (above n10 n16)
(above n11 n12) (above n11 n13) (above n11 n14) (above n11 n15) (above n11 n16)
(above n12 n13) (above n12 n14) (above n12 n15) (above n12 n16)
(above n13 n14) (above n13 n15) (above n13 n16)
(above n14 n15) (above n14 n16)
(above n15 n16)

(lift-at fast0 n8)
(passengers fast0 n0)
(can-hold fast0 n1) (can-hold fast0 n2) (can-hold fast0 n3) (can-hold fast0 n4)
(reachable-floor fast0 n0)(reachable-floor fast0 n4)(reachable-floor fast0 n8)(reachable-floor fast0 n12)(reachable-floor fast0 n16)

(lift-at fast1 n12)
(passengers fast1 n0)
(can-hold fast1 n1) (can-hold fast1 n2) (can-hold fast1 n3) (can-hold fast1 n4)
(reachable-floor fast1 n0)(reachable-floor fast1 n4)(reachable-floor fast1 n8)(reachable-floor fast1 n12)(reachable-floor fast1 n16)

(lift-at slow0-0 n2)
(passengers slow0-0 n0)
(can-hold slow0-0 n1) (can-hold slow0-0 n2) (can-hold slow0-0 n3)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0 n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0 n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0 n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0 n7)(reachable-floor slow0-0 n8)

(lift-at slow1-0 n12)
(passengers slow1-0 n0)
(can-hold slow1-0 n1) (can-hold slow1-0 n2) (can-hold slow1-0 n3)
(reachable-floor slow1-0 n8)(reachable-floor slow1-0 n9)(reachable-floor slow1-0 n10)(reachable-floor slow1-0 n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0 n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0 n15)(reachable-floor slow1-0 n16)

(passenger-at p0 n13)
(passenger-at p1 n10)

```
(passenger-at p2 n13)
(passenger-at p3 n0)
(passenger-at p4 n9)
(passenger-at p5 n12)
(passenger-at p6 n8)
(passenger-at p7 n3)
(passenger-at p8 n5)
(passenger-at p9 n2)
(passenger-at p10 n4)
(passenger-at p11 n11)
(passenger-at p12 n13)
(passenger-at p13 n7)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (=
(travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-
slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow
n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n1 n2)
6) (= (travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (=
(travel-slow n1 n5) 9) (= (travel-slow n1 n6) 10) (=
(travel-slow n1 n7) 11) (= (travel-slow n1 n8) 12) (=
(travel-slow n2 n3) 6) (= (travel-slow n2 n4) 7) (= (travel-
slow n2 n5) 8) (= (travel-slow n2 n6) 9) (= (travel-slow n2
n7) 10) (= (travel-slow n2 n8) 11) (= (travel-slow n3 n4) 6)
(= (travel-slow n3 n5) 7) (= (travel-slow n3 n6) 8) (=
(travel-slow n3 n7) 9) (= (travel-slow n3 n8) 10) (=
(travel-slow n4 n5) 6) (= (travel-slow n4 n6) 7) (= (travel-
slow n4 n7) 8) (= (travel-slow n4 n8) 9) (= (travel-slow n5
n6) 6) (= (travel-slow n5 n7) 7) (= (travel-slow n5 n8) 8)
(= (travel-slow n6 n7) 6) (= (travel-slow n6 n8) 7) (=
(travel-slow n7 n8) 6)

(= (travel-slow n8 n9) 6) (= (travel-slow n8 n10) 7) (=
(travel-slow n8 n11) 8) (= (travel-slow n8 n12) 9) (=
(travel-slow n8 n13) 10) (= (travel-slow n8 n14) 11) (=
(travel-slow n8 n15) 12) (= (travel-slow n8 n16) 13) (=
(travel-slow n9 n10) 6) (= (travel-slow n9 n11) 7) (=
(travel-slow n9 n12) 8) (= (travel-slow n9 n13) 9) (=
(travel-slow n9 n14) 10) (= (travel-slow n9 n15) 11) (=
(travel-slow n9 n16) 12) (= (travel-slow n10 n11) 6) (=
(travel-slow n10 n12) 7) (= (travel-slow n10 n13) 8) (=
(travel-slow n10 n14) 9) (= (travel-slow n10 n15) 10) (=
(travel-slow n10 n16) 11) (= (travel-slow n11 n12) 6) (=
(travel-slow n11 n13) 7) (= (travel-slow n11 n14) 8) (=
(travel-slow n11 n15) 9) (= (travel-slow n11 n16) 10) (=
(travel-slow n12 n13) 6) (= (travel-slow n12 n14) 7) (=
(travel-slow n12 n15) 8) (= (travel-slow n12 n16) 9) (=
(travel-slow n13 n14) 6) (= (travel-slow n13 n15) 7) (=
(travel-slow n13 n16) 8) (= (travel-slow n14 n15) 6) (=
(travel-slow n14 n16) 7) (= (travel-slow n15 n16) 6)
```

```
(= (travel-fast n0 n4) 13) (= (travel-fast n0 n8) 25) (=
(travel-fast n0 n12) 37) (= (travel-fast n0 n16) 49)

(= (travel-fast n4 n8) 13) (= (travel-fast n4 n12) 25) (=
(travel-fast n4 n16) 37)

(= (travel-fast n8 n12) 13) (= (travel-fast n8 n16) 25)

(= (travel-fast n12 n16) 13)

(= (total-cost) 0)

)

(:goal
(and
(passenger-at p0 n8)
(passenger-at p1 n15)
(passenger-at p2 n6)
(passenger-at p3 n14)
(passenger-at p4 n5)
(passenger-at p5 n2)
(passenger-at p6 n14)
(passenger-at p7 n4)
(passenger-at p8 n10)
(passenger-at p9 n9)
(passenger-at p10 n12)
(passenger-at p11 n13)
(passenger-at p12 n5)
(passenger-at p13 n6)
))

(:metric minimize (total-cost))

)
```

8.2.3 Problema 20

```
(define (problem elevators-sequencedstrips-p40_60_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16
n17 n18 n19 n20 n21 n22 n23 n24 n25 n26 n27 n28 n29 n30 n31
n32 n33 n34 n35 n36 n37 n38 n39 n40  - count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16
p17 p18 p19 p20 p21 p22 p23 p24 p25 p26 p27 p28 p29 p30 p31
p32 p33 p34 p35 p36 p37 p38 p39 p40 p41 p42 p43 p44 p45 p46
p47 p48 p49 p50 p51 p52 p53 p54 p55 p56 p57 p58 p59   -
passenger
fast0 fast1 fast2 fast3  - fast-elevator
slow0-0 slow1-0 slow2-0 slow3-0 - slow-elevator
)

(:init
(next n0 n1) (next n1 n2) (next n2 n3) (next n3 n4) (next n4
n5) (next n5 n6) (next n6 n7) (next n7 n8) (next n8 n9)
(next n9 n10) (next n10 n11) (next n11 n12) (next n12 n13)
(next n13 n14) (next n14 n15) (next n15 n16) (next n16 n17)
(next n17 n18) (next n18 n19) (next n19 n20) (next n20 n21)
(next n21 n22) (next n22 n23) (next n23 n24) (next n24 n25)
(next n25 n26) (next n26 n27) (next n27 n28) (next n28 n29)
(next n29 n30) (next n30 n31) (next n31 n32) (next n32 n33)
(next n33 n34) (next n34 n35) (next n35 n36) (next n36 n37)
(next n37 n38) (next n38 n39) (next n39 n40)

(above n0 n1) (above n0 n2) (above n0 n3) (above n0 n4)
(above n0 n5) (above n0 n6) (above n0 n7) (above n0 n8)
(above n0 n9) (above n0 n10) (above n0 n11) (above n0 n12)
(above n0 n13) (above n0 n14) (above n0 n15) (above n0 n16)
(above n0 n17) (above n0 n18) (above n0 n19) (above n0 n20)
(above n0 n21) (above n0 n22) (above n0 n23) (above n0 n24)
(above n0 n25) (above n0 n26) (above n0 n27) (above n0 n28)
(above n0 n29) (above n0 n30) (above n0 n31) (above n0 n32)
(above n0 n33) (above n0 n34) (above n0 n35) (above n0 n36)
(above n0 n37) (above n0 n38) (above n0 n39) (above n0 n40)
(above n1 n2) (above n1 n3) (above n1 n4) (above n1 n5)
(above n1 n6) (above n1 n7) (above n1 n8) (above n1 n9)
(above n1 n10) (above n1 n11) (above n1 n12) (above n1 n13)
(above n1 n14) (above n1 n15) (above n1 n16) (above n1 n17)
(above n1 n18) (above n1 n19) (above n1 n20) (above n1 n21)
(above n1 n22) (above n1 n23) (above n1 n24) (above n1 n25)
(above n1 n26) (above n1 n27) (above n1 n28) (above n1 n29)
(above n1 n30) (above n1 n31) (above n1 n32) (above n1 n33)
```

```
(above n1 n34) (above n1 n35) (above n1 n36) (above n1 n37)
(above n1 n38) (above n1 n39) (above n1 n40)
(above n2 n3) (above n2 n4) (above n2 n5) (above n2 n6)
(above n2 n7) (above n2 n8) (above n2 n9) (above n2 n10)
(above n2 n11) (above n2 n12) (above n2 n13) (above n2 n14)
(above n2 n15) (above n2 n16) (above n2 n17) (above n2 n18)
(above n2 n19) (above n2 n20) (above n2 n21) (above n2 n22)
(above n2 n23) (above n2 n24) (above n2 n25) (above n2 n26)
(above n2 n27) (above n2 n28) (above n2 n29) (above n2 n30)
(above n2 n31) (above n2 n32) (above n2 n33) (above n2 n34)
(above n2 n35) (above n2 n36) (above n2 n37) (above n2 n38)
(above n2 n39) (above n2 n40)
(above n3 n4) (above n3 n5) (above n3 n6) (above n3 n7)
(above n3 n8) (above n3 n9) (above n3 n10) (above n3 n11)
(above n3 n12) (above n3 n13) (above n3 n14) (above n3 n15)
(above n3 n16) (above n3 n17) (above n3 n18) (above n3 n19)
(above n3 n20) (above n3 n21) (above n3 n22) (above n3 n23)
(above n3 n24) (above n3 n25) (above n3 n26) (above n3 n27)
(above n3 n28) (above n3 n29) (above n3 n30) (above n3 n31)
(above n3 n32) (above n3 n33) (above n3 n34) (above n3 n35)
(above n3 n36) (above n3 n37) (above n3 n38) (above n3 n39)
(above n3 n40)
(above n4 n5) (above n4 n6) (above n4 n7) (above n4 n8)
(above n4 n9) (above n4 n10) (above n4 n11) (above n4 n12)
(above n4 n13) (above n4 n14) (above n4 n15) (above n4 n16)
(above n4 n17) (above n4 n18) (above n4 n19) (above n4 n20)
(above n4 n21) (above n4 n22) (above n4 n23) (above n4 n24)
(above n4 n25) (above n4 n26) (above n4 n27) (above n4 n28)
(above n4 n29) (above n4 n30) (above n4 n31) (above n4 n32)
(above n4 n33) (above n4 n34) (above n4 n35) (above n4 n36)
(above n4 n37) (above n4 n38) (above n4 n39) (above n4 n40)
(above n5 n6) (above n5 n7) (above n5 n8) (above n5 n9)
(above n5 n10) (above n5 n11) (above n5 n12) (above n5 n13)
(above n5 n14) (above n5 n15) (above n5 n16) (above n5 n17)
(above n5 n18) (above n5 n19) (above n5 n20) (above n5 n21)
(above n5 n22) (above n5 n23) (above n5 n24) (above n5 n25)
(above n5 n26) (above n5 n27) (above n5 n28) (above n5 n29)
(above n5 n30) (above n5 n31) (above n5 n32) (above n5 n33)
(above n5 n34) (above n5 n35) (above n5 n36) (above n5 n37)
(above n5 n38) (above n5 n39) (above n5 n40)
(above n6 n7) (above n6 n8) (above n6 n9) (above n6 n10)
(above n6 n11) (above n6 n12) (above n6 n13) (above n6 n14)
(above n6 n15) (above n6 n16) (above n6 n17) (above n6 n18)
(above n6 n19) (above n6 n20) (above n6 n21) (above n6 n22)
(above n6 n23) (above n6 n24) (above n6 n25) (above n6 n26)
(above n6 n27) (above n6 n28) (above n6 n29) (above n6 n30)
(above n6 n31) (above n6 n32) (above n6 n33) (above n6 n34)
(above n6 n35) (above n6 n36) (above n6 n37) (above n6 n38)
(above n6 n39) (above n6 n40)
```

(above n7 n8) (above n7 n9) (above n7 n10) (above n7 n11) (above n7 n12) (above n7 n13) (above n7 n14) (above n7 n15) (above n7 n16) (above n7 n17) (above n7 n18) (above n7 n19) (above n7 n20) (above n7 n21) (above n7 n22) (above n7 n23) (above n7 n24) (above n7 n25) (above n7 n26) (above n7 n27) (above n7 n28) (above n7 n29) (above n7 n30) (above n7 n31) (above n7 n32) (above n7 n33) (above n7 n34) (above n7 n35) (above n7 n36) (above n7 n37) (above n7 n38) (above n7 n39) (above n7 n40)
(above n8 n9) (above n8 n10) (above n8 n11) (above n8 n12) (above n8 n13) (above n8 n14) (above n8 n15) (above n8 n16) (above n8 n17) (above n8 n18) (above n8 n19) (above n8 n20) (above n8 n21) (above n8 n22) (above n8 n23) (above n8 n24) (above n8 n25) (above n8 n26) (above n8 n27) (above n8 n28) (above n8 n29) (above n8 n30) (above n8 n31) (above n8 n32) (above n8 n33) (above n8 n34) (above n8 n35) (above n8 n36) (above n8 n37) (above n8 n38) (above n8 n39) (above n8 n40)
(above n9 n10) (above n9 n11) (above n9 n12) (above n9 n13) (above n9 n14) (above n9 n15) (above n9 n16) (above n9 n17) (above n9 n18) (above n9 n19) (above n9 n20) (above n9 n21) (above n9 n22) (above n9 n23) (above n9 n24) (above n9 n25) (above n9 n26) (above n9 n27) (above n9 n28) (above n9 n29) (above n9 n30) (above n9 n31) (above n9 n32) (above n9 n33) (above n9 n34) (above n9 n35) (above n9 n36) (above n9 n37) (above n9 n38) (above n9 n39) (above n9 n40)
(above n10 n11) (above n10 n12) (above n10 n13) (above n10 n14) (above n10 n15) (above n10 n16) (above n10 n17) (above n10 n18) (above n10 n19) (above n10 n20) (above n10 n21) (above n10 n22) (above n10 n23) (above n10 n24) (above n10 n25) (above n10 n26) (above n10 n27) (above n10 n28) (above n10 n29) (above n10 n30) (above n10 n31) (above n10 n32) (above n10 n33) (above n10 n34) (above n10 n35) (above n10 n36) (above n10 n37) (above n10 n38) (above n10 n39) (above n10 n40)
(above n11 n12) (above n11 n13) (above n11 n14) (above n11 n15) (above n11 n16) (above n11 n17) (above n11 n18) (above n11 n19) (above n11 n20) (above n11 n21) (above n11 n22) (above n11 n23) (above n11 n24) (above n11 n25) (above n11 n26) (above n11 n27) (above n11 n28) (above n11 n29) (above n11 n30) (above n11 n31) (above n11 n32) (above n11 n33) (above n11 n34) (above n11 n35) (above n11 n36) (above n11 n37) (above n11 n38) (above n11 n39) (above n11 n40)
(above n12 n13) (above n12 n14) (above n12 n15) (above n12 n16) (above n12 n17) (above n12 n18) (above n12 n19) (above n12 n20) (above n12 n21) (above n12 n22) (above n12 n23) (above n12 n24) (above n12 n25) (above n12 n26) (above n12 n27) (above n12 n28) (above n12 n29) (above n12 n30) (above n12 n31) (above n12 n32) (above n12 n33) (above n12 n34) (above n12 n35) (above n12 n36) (above n12 n37) (above n12 n38) (above n12 n39) (above n12 n40)

```
(above n13 n14) (above n13 n15) (above n13 n16) (above n13
n17) (above n13 n18) (above n13 n19) (above n13 n20) (above
n13 n21) (above n13 n22) (above n13 n23) (above n13 n24)
(above n13 n25) (above n13 n26) (above n13 n27) (above n13
n28) (above n13 n29) (above n13 n30) (above n13 n31) (above
n13 n32) (above n13 n33) (above n13 n34) (above n13 n35)
(above n13 n36) (above n13 n37) (above n13 n38) (above n13
n39) (above n13 n40)
(above n14 n15) (above n14 n16) (above n14 n17) (above n14
n18) (above n14 n19) (above n14 n20) (above n14 n21) (above
n14 n22) (above n14 n23) (above n14 n24) (above n14 n25)
(above n14 n26) (above n14 n27) (above n14 n28) (above n14
n29) (above n14 n30) (above n14 n31) (above n14 n32) (above
n14 n33) (above n14 n34) (above n14 n35) (above n14 n36)
(above n14 n37) (above n14 n38) (above n14 n39) (above n14
n40)
(above n15 n16) (above n15 n17) (above n15 n18) (above n15
n19) (above n15 n20) (above n15 n21) (above n15 n22) (above
n15 n23) (above n15 n24) (above n15 n25) (above n15 n26)
(above n15 n27) (above n15 n28) (above n15 n29) (above n15
n30) (above n15 n31) (above n15 n32) (above n15 n33) (above
n15 n34) (above n15 n35) (above n15 n36) (above n15 n37)
(above n15 n38) (above n15 n39) (above n15 n40)
(above n16 n17) (above n16 n18) (above n16 n19) (above n16
n20) (above n16 n21) (above n16 n22) (above n16 n23) (above
n16 n24) (above n16 n25) (above n16 n26) (above n16 n27)
(above n16 n28) (above n16 n29) (above n16 n30) (above n16
n31) (above n16 n32) (above n16 n33) (above n16 n34) (above
n16 n35) (above n16 n36) (above n16 n37) (above n16 n38)
(above n16 n39) (above n16 n40)
(above n17 n18) (above n17 n19) (above n17 n20) (above n17
n21) (above n17 n22) (above n17 n23) (above n17 n24) (above
n17 n25) (above n17 n26) (above n17 n27) (above n17 n28)
(above n17 n29) (above n17 n30) (above n17 n31) (above n17
n32) (above n17 n33) (above n17 n34) (above n17 n35) (above
n17 n36) (above n17 n37) (above n17 n38) (above n17 n39)
(above n17 n40)
(above n18 n19) (above n18 n20) (above n18 n21) (above n18
n22) (above n18 n23) (above n18 n24) (above n18 n25) (above
n18 n26) (above n18 n27) (above n18 n28) (above n18 n29)
(above n18 n30) (above n18 n31) (above n18 n32) (above n18
n33) (above n18 n34) (above n18 n35) (above n18 n36) (above
n18 n37) (above n18 n38) (above n18 n39) (above n18 n40)
(above n19 n20) (above n19 n21) (above n19 n22) (above n19
n23) (above n19 n24) (above n19 n25) (above n19 n26) (above
n19 n27) (above n19 n28) (above n19 n29) (above n19 n30)
(above n19 n31) (above n19 n32) (above n19 n33) (above n19
n34) (above n19 n35) (above n19 n36) (above n19 n37) (above
n19 n38) (above n19 n39) (above n19 n40)
```

(above n20 n21) (above n20 n22) (above n20 n23) (above n20 n24) (above n20 n25) (above n20 n26) (above n20 n27) (above n20 n28) (above n20 n29) (above n20 n30) (above n20 n31) (above n20 n32) (above n20 n33) (above n20 n34) (above n20 n35) (above n20 n36) (above n20 n37) (above n20 n38) (above n20 n39) (above n20 n40)
(above n21 n22) (above n21 n23) (above n21 n24) (above n21 n25) (above n21 n26) (above n21 n27) (above n21 n28) (above n21 n29) (above n21 n30) (above n21 n31) (above n21 n32) (above n21 n33) (above n21 n34) (above n21 n35) (above n21 n36) (above n21 n37) (above n21 n38) (above n21 n39) (above n21 n40)
(above n22 n23) (above n22 n24) (above n22 n25) (above n22 n26) (above n22 n27) (above n22 n28) (above n22 n29) (above n22 n30) (above n22 n31) (above n22 n32) (above n22 n33) (above n22 n34) (above n22 n35) (above n22 n36) (above n22 n37) (above n22 n38) (above n22 n39) (above n22 n40)
(above n23 n24) (above n23 n25) (above n23 n26) (above n23 n27) (above n23 n28) (above n23 n29) (above n23 n30) (above n23 n31) (above n23 n32) (above n23 n33) (above n23 n34) (above n23 n35) (above n23 n36) (above n23 n37) (above n23 n38) (above n23 n39) (above n23 n40)
(above n24 n25) (above n24 n26) (above n24 n27) (above n24 n28) (above n24 n29) (above n24 n30) (above n24 n31) (above n24 n32) (above n24 n33) (above n24 n34) (above n24 n35) (above n24 n36) (above n24 n37) (above n24 n38) (above n24 n39) (above n24 n40)
(above n25 n26) (above n25 n27) (above n25 n28) (above n25 n29) (above n25 n30) (above n25 n31) (above n25 n32) (above n25 n33) (above n25 n34) (above n25 n35) (above n25 n36) (above n25 n37) (above n25 n38) (above n25 n39) (above n25 n40)
(above n26 n27) (above n26 n28) (above n26 n29) (above n26 n30) (above n26 n31) (above n26 n32) (above n26 n33) (above n26 n34) (above n26 n35) (above n26 n36) (above n26 n37) (above n26 n38) (above n26 n39) (above n26 n40)
(above n27 n28) (above n27 n29) (above n27 n30) (above n27 n31) (above n27 n32) (above n27 n33) (above n27 n34) (above n27 n35) (above n27 n36) (above n27 n37) (above n27 n38) (above n27 n39) (above n27 n40)
(above n28 n29) (above n28 n30) (above n28 n31) (above n28 n32) (above n28 n33) (above n28 n34) (above n28 n35) (above n28 n36) (above n28 n37) (above n28 n38) (above n28 n39) (above n28 n40)
(above n29 n30) (above n29 n31) (above n29 n32) (above n29 n33) (above n29 n34) (above n29 n35) (above n29 n36) (above n29 n37) (above n29 n38) (above n29 n39) (above n29 n40)
(above n30 n31) (above n30 n32) (above n30 n33) (above n30 n34) (above n30 n35) (above n30 n36) (above n30 n37) (above n30 n38) (above n30 n39) (above n30 n40)

(above n31 n32) (above n31 n33) (above n31 n34) (above n31 n35) (above n31 n36) (above n31 n37) (above n31 n38) (above n31 n39) (above n31 n40)
(above n32 n33) (above n32 n34) (above n32 n35) (above n32 n36) (above n32 n37) (above n32 n38) (above n32 n39) (above n32 n40)
(above n33 n34) (above n33 n35) (above n33 n36) (above n33 n37) (above n33 n38) (above n33 n39) (above n33 n40)
(above n34 n35) (above n34 n36) (above n34 n37) (above n34 n38) (above n34 n39) (above n34 n40)
(above n35 n36) (above n35 n37) (above n35 n38) (above n35 n39) (above n35 n40)
(above n36 n37) (above n36 n38) (above n36 n39) (above n36 n40)
(above n37 n38) (above n37 n39) (above n37 n40)
(above n38 n39) (above n38 n40)
(above n39 n40)

(lift-at fast0 n10)
(passengers fast0 n0)
(can-hold fast0 n1) (can-hold fast0 n2) (can-hold fast0 n3)
(can-hold fast0 n4) (can-hold fast0 n5) (can-hold fast0 n6)
(reachable-floor fast0 n0)(reachable-floor fast0 n5)(reachable-floor fast0 n10)(reachable-floor fast0 n15)(reachable-floor fast0 n20)(reachable-floor fast0 n25)(reachable-floor fast0 n30)(reachable-floor fast0 n35)(reachable-floor fast0 n40)

(lift-at fast1 n40)
(passengers fast1 n0)
(can-hold fast1 n1) (can-hold fast1 n2) (can-hold fast1 n3)
(can-hold fast1 n4) (can-hold fast1 n5) (can-hold fast1 n6)
(reachable-floor fast1 n0)(reachable-floor fast1 n5)(reachable-floor fast1 n10)(reachable-floor fast1 n15)(reachable-floor fast1 n20)(reachable-floor fast1 n25)(reachable-floor fast1 n30)(reachable-floor fast1 n35)(reachable-floor fast1 n40)

(lift-at fast2 n0)
(passengers fast2 n0)
(can-hold fast2 n1) (can-hold fast2 n2) (can-hold fast2 n3)
(can-hold fast2 n4) (can-hold fast2 n5) (can-hold fast2 n6)
(reachable-floor fast2 n0)(reachable-floor fast2 n5)(reachable-floor fast2 n10)(reachable-floor fast2 n15)(reachable-floor fast2 n20)(reachable-floor fast2 n25)(reachable-floor fast2 n30)(reachable-floor fast2 n35)(reachable-floor fast2 n40)

```
(lift-at fast3 n20)
(passengers fast3 n0)
(can-hold fast3 n1) (can-hold fast3 n2) (can-hold fast3 n3)
(can-hold fast3 n4) (can-hold fast3 n5) (can-hold fast3 n6)
(reachable-floor fast3 n0)(reachable-floor fast3
n5)(reachable-floor fast3 n10)(reachable-floor fast3
n15)(reachable-floor fast3 n20)(reachable-floor fast3
n25)(reachable-floor fast3 n30)(reachable-floor fast3
n35)(reachable-floor fast3 n40)

(lift-at slow0-0 n9)
(passengers slow0-0 n0)
(can-hold slow0-0 n1) (can-hold slow0-0 n2) (can-hold slow0-
0 n3) (can-hold slow0-0 n4)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0
n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0
n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0
n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0
n7)(reachable-floor slow0-0 n8)(reachable-floor slow0-0
n9)(reachable-floor slow0-0 n10)

(lift-at slow1-0 n12)
(passengers slow1-0 n0)
(can-hold slow1-0 n1) (can-hold slow1-0 n2) (can-hold slow1-
0 n3) (can-hold slow1-0 n4)
(reachable-floor slow1-0 n10)(reachable-floor slow1-0
n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0
n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0
n15)(reachable-floor slow1-0 n16)(reachable-floor slow1-0
n17)(reachable-floor slow1-0 n18)(reachable-floor slow1-0
n19)(reachable-floor slow1-0 n20)

(lift-at slow2-0 n23)
(passengers slow2-0 n0)
(can-hold slow2-0 n1) (can-hold slow2-0 n2) (can-hold slow2-
0 n3) (can-hold slow2-0 n4)
(reachable-floor slow2-0 n20)(reachable-floor slow2-0
n21)(reachable-floor slow2-0 n22)(reachable-floor slow2-0
n23)(reachable-floor slow2-0 n24)(reachable-floor slow2-0
n25)(reachable-floor slow2-0 n26)(reachable-floor slow2-0
n27)(reachable-floor slow2-0 n28)(reachable-floor slow2-0
n29)(reachable-floor slow2-0 n30)

(lift-at slow3-0 n36)
(passengers slow3-0 n0)
(can-hold slow3-0 n1) (can-hold slow3-0 n2) (can-hold slow3-
0 n3) (can-hold slow3-0 n4)
(reachable-floor slow3-0 n30)(reachable-floor slow3-0
n31)(reachable-floor slow3-0 n32)(reachable-floor slow3-0
n33)(reachable-floor slow3-0 n34)(reachable-floor slow3-0
```

```
n35)(reachable-floor slow3-0 n36)(reachable-floor slow3-0
n37)(reachable-floor slow3-0 n38)(reachable-floor slow3-0
n39)(reachable-floor slow3-0 n40)

(passenger-at p0 n18)
(passenger-at p1 n32)
(passenger-at p2 n17)
(passenger-at p3 n32)
(passenger-at p4 n35)
(passenger-at p5 n3)
(passenger-at p6 n38)
(passenger-at p7 n38)
(passenger-at p8 n36)
(passenger-at p9 n19)
(passenger-at p10 n34)
(passenger-at p11 n19)
(passenger-at p12 n2)
(passenger-at p13 n25)
(passenger-at p14 n13)
(passenger-at p15 n29)
(passenger-at p16 n1)
(passenger-at p17 n11)
(passenger-at p18 n9)
(passenger-at p19 n19)
(passenger-at p20 n8)
(passenger-at p21 n11)
(passenger-at p22 n17)
(passenger-at p23 n11)
(passenger-at p24 n21)
(passenger-at p25 n10)
(passenger-at p26 n39)
(passenger-at p27 n24)
(passenger-at p28 n25)
(passenger-at p29 n9)
(passenger-at p30 n22)
(passenger-at p31 n37)
(passenger-at p32 n15)
(passenger-at p33 n18)
(passenger-at p34 n12)
(passenger-at p35 n38)
(passenger-at p36 n17)
(passenger-at p37 n36)
(passenger-at p38 n10)
(passenger-at p39 n10)
(passenger-at p40 n27)
(passenger-at p41 n12)
(passenger-at p42 n20)
(passenger-at p43 n23)
(passenger-at p44 n29)
(passenger-at p45 n40)
```

```
(passenger-at p46 n16)
(passenger-at p47 n36)
(passenger-at p48 n2)
(passenger-at p49 n16)
(passenger-at p50 n11)
(passenger-at p51 n35)
(passenger-at p52 n23)
(passenger-at p53 n22)
(passenger-at p54 n38)
(passenger-at p55 n24)
(passenger-at p56 n13)
(passenger-at p57 n10)
(passenger-at p58 n32)
(passenger-at p59 n38)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (=
(travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-
slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow
n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n0 n9)
14) (= (travel-slow n0 n10) 15) (= (travel-slow n1 n2) 6) (=
(travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (= (travel-
slow n1 n5) 9) (= (travel-slow n1 n6) 10) (= (travel-slow n1
n7) 11) (= (travel-slow n1 n8) 12) (= (travel-slow n1 n9)
13) (= (travel-slow n1 n10) 14) (= (travel-slow n2 n3) 6) (=
(travel-slow n2 n4) 7) (= (travel-slow n2 n5) 8) (= (travel-
slow n2 n6) 9) (= (travel-slow n2 n7) 10) (= (travel-slow n2
n8) 11) (= (travel-slow n2 n9) 12) (= (travel-slow n2 n10)
13) (= (travel-slow n3 n4) 6) (= (travel-slow n3 n5) 7) (=
(travel-slow n3 n6) 8) (= (travel-slow n3 n7) 9) (= (travel-
slow n3 n8) 10) (= (travel-slow n3 n9) 11) (= (travel-slow
n3 n10) 12) (= (travel-slow n4 n5) 6) (= (travel-slow n4 n6)
7) (= (travel-slow n4 n7) 8) (= (travel-slow n4 n8) 9) (=
(travel-slow n4 n9) 10) (= (travel-slow n4 n10) 11) (=
(travel-slow n5 n6) 6) (= (travel-slow n5 n7) 7) (= (travel-
slow n5 n8) 8) (= (travel-slow n5 n9) 9) (= (travel-slow n5
n10) 10) (= (travel-slow n6 n7) 6) (= (travel-slow n6 n8) 7)
(= (travel-slow n6 n9) 8) (= (travel-slow n6 n10) 9) (=
(travel-slow n7 n8) 6) (= (travel-slow n7 n9) 7) (= (travel-
slow n7 n10) 8) (= (travel-slow n8 n9) 6) (= (travel-slow n8
n10) 7) (= (travel-slow n9 n10) 6)

(= (travel-slow n10 n11) 6) (= (travel-slow n10 n12) 7) (=
(travel-slow n10 n13) 8) (= (travel-slow n10 n14) 9) (=
(travel-slow n10 n15) 10) (= (travel-slow n10 n16) 11) (=
(travel-slow n10 n17) 12) (= (travel-slow n10 n18) 13) (=
(travel-slow n10 n19) 14) (= (travel-slow n10 n20) 15) (=
(travel-slow n11 n12) 6) (= (travel-slow n11 n13) 7) (=
(travel-slow n11 n14) 8) (= (travel-slow n11 n15) 9) (=
(travel-slow n11 n16) 10) (= (travel-slow n11 n17) 11) (=
(travel-slow n11 n18) 12) (= (travel-slow n11 n19) 13) (=
```

```
(travel-slow n11 n20) 14) (= (travel-slow n12 n13) 6) (=
(travel-slow n12 n14) 7) (= (travel-slow n12 n15) 8) (=
(travel-slow n12 n16) 9) (= (travel-slow n12 n17) 10) (=
(travel-slow n12 n18) 11) (= (travel-slow n12 n19) 12) (=
(travel-slow n12 n20) 13) (= (travel-slow n13 n14) 6) (=
(travel-slow n13 n15) 7) (= (travel-slow n13 n16) 8) (=
(travel-slow n13 n17) 9) (= (travel-slow n13 n18) 10) (=
(travel-slow n13 n19) 11) (= (travel-slow n13 n20) 12) (=
(travel-slow n14 n15) 6) (= (travel-slow n14 n16) 7) (=
(travel-slow n14 n17) 8) (= (travel-slow n14 n18) 9) (=
(travel-slow n14 n19) 10) (= (travel-slow n14 n20) 11) (=
(travel-slow n15 n16) 6) (= (travel-slow n15 n17) 7) (=
(travel-slow n15 n18) 8) (= (travel-slow n15 n19) 9) (=
(travel-slow n15 n20) 10) (= (travel-slow n16 n17) 6) (=
(travel-slow n16 n18) 7) (= (travel-slow n16 n19) 8) (=
(travel-slow n16 n20) 9) (= (travel-slow n17 n18) 6) (=
(travel-slow n17 n19) 7) (= (travel-slow n17 n20) 8) (=
(travel-slow n18 n19) 6) (= (travel-slow n18 n20) 7) (=
(travel-slow n19 n20) 6)

(= (travel-slow n20 n21) 6) (= (travel-slow n20 n22) 7) (=
(travel-slow n20 n23) 8) (= (travel-slow n20 n24) 9) (=
(travel-slow n20 n25) 10) (= (travel-slow n20 n26) 11) (=
(travel-slow n20 n27) 12) (= (travel-slow n20 n28) 13) (=
(travel-slow n20 n29) 14) (= (travel-slow n20 n30) 15) (=
(travel-slow n21 n22) 6) (= (travel-slow n21 n23) 7) (=
(travel-slow n21 n24) 8) (= (travel-slow n21 n25) 9) (=
(travel-slow n21 n26) 10) (= (travel-slow n21 n27) 11) (=
(travel-slow n21 n28) 12) (= (travel-slow n21 n29) 13) (=
(travel-slow n21 n30) 14) (= (travel-slow n22 n23) 6) (=
(travel-slow n22 n24) 7) (= (travel-slow n22 n25) 8) (=
(travel-slow n22 n26) 9) (= (travel-slow n22 n27) 10) (=
(travel-slow n22 n28) 11) (= (travel-slow n22 n29) 12) (=
(travel-slow n22 n30) 13) (= (travel-slow n23 n24) 6) (=
(travel-slow n23 n25) 7) (= (travel-slow n23 n26) 8) (=
(travel-slow n23 n27) 9) (= (travel-slow n23 n28) 10) (=
(travel-slow n23 n29) 11) (= (travel-slow n23 n30) 12) (=
(travel-slow n24 n25) 6) (= (travel-slow n24 n26) 7) (=
(travel-slow n24 n27) 8) (= (travel-slow n24 n28) 9) (=
(travel-slow n24 n29) 10) (= (travel-slow n24 n30) 11) (=
(travel-slow n25 n26) 6) (= (travel-slow n25 n27) 7) (=
(travel-slow n25 n28) 8) (= (travel-slow n25 n29) 9) (=
(travel-slow n25 n30) 10) (= (travel-slow n26 n27) 6) (=
(travel-slow n26 n28) 7) (= (travel-slow n26 n29) 8) (=
(travel-slow n26 n30) 9) (= (travel-slow n27 n28) 6) (=
(travel-slow n27 n29) 7) (= (travel-slow n27 n30) 8) (=
(travel-slow n28 n29) 6) (= (travel-slow n28 n30) 7) (=
(travel-slow n29 n30) 6)
```

```
(= (travel-slow n30 n31) 6) (= (travel-slow n30 n32) 7) (=
(travel-slow n30 n33) 8) (= (travel-slow n30 n34) 9) (=
(travel-slow n30 n35) 10) (= (travel-slow n30 n36) 11) (=
(travel-slow n30 n37) 12) (= (travel-slow n30 n38) 13) (=
(travel-slow n30 n39) 14) (= (travel-slow n30 n40) 15) (=
(travel-slow n31 n32) 6) (= (travel-slow n31 n33) 7) (=
(travel-slow n31 n34) 8) (= (travel-slow n31 n35) 9) (=
(travel-slow n31 n36) 10) (= (travel-slow n31 n37) 11) (=
(travel-slow n31 n38) 12) (= (travel-slow n31 n39) 13) (=
(travel-slow n31 n40) 14) (= (travel-slow n32 n33) 6) (=
(travel-slow n32 n34) 7) (= (travel-slow n32 n35) 8) (=
(travel-slow n32 n36) 9) (= (travel-slow n32 n37) 10) (=
(travel-slow n32 n38) 11) (= (travel-slow n32 n39) 12) (=
(travel-slow n32 n40) 13) (= (travel-slow n33 n34) 6) (=
(travel-slow n33 n35) 7) (= (travel-slow n33 n36) 8) (=
(travel-slow n33 n37) 9) (= (travel-slow n33 n38) 10) (=
(travel-slow n33 n39) 11) (= (travel-slow n33 n40) 12) (=
(travel-slow n34 n35) 6) (= (travel-slow n34 n36) 7) (=
(travel-slow n34 n37) 8) (= (travel-slow n34 n38) 9) (=
(travel-slow n34 n39) 10) (= (travel-slow n34 n40) 11) (=
(travel-slow n35 n36) 6) (= (travel-slow n35 n37) 7) (=
(travel-slow n35 n38) 8) (= (travel-slow n35 n39) 9) (=
(travel-slow n35 n40) 10) (= (travel-slow n36 n37) 6) (=
(travel-slow n36 n38) 7) (= (travel-slow n36 n39) 8) (=
(travel-slow n36 n40) 9) (= (travel-slow n37 n38) 6) (=
(travel-slow n37 n39) 7) (= (travel-slow n37 n40) 8) (=
(travel-slow n38 n39) 6) (= (travel-slow n38 n40) 7) (=
(travel-slow n39 n40) 6)


(= (travel-fast n0 n5) 16) (= (travel-fast n0 n10) 31) (=
(travel-fast n0 n15) 46) (= (travel-fast n0 n20) 61) (=
(travel-fast n0 n25) 76) (= (travel-fast n0 n30) 91) (=
(travel-fast n0 n35) 106) (= (travel-fast n0 n40) 121)

(= (travel-fast n5 n10) 16) (= (travel-fast n5 n15) 31) (=
(travel-fast n5 n20) 46) (= (travel-fast n5 n25) 61) (=
(travel-fast n5 n30) 76) (= (travel-fast n5 n35) 91) (=
(travel-fast n5 n40) 106)

(= (travel-fast n10 n15) 16) (= (travel-fast n10 n20) 31) (=
(travel-fast n10 n25) 46) (= (travel-fast n10 n30) 61) (=
(travel-fast n10 n35) 76) (= (travel-fast n10 n40) 91)

(= (travel-fast n15 n20) 16) (= (travel-fast n15 n25) 31) (=
(travel-fast n15 n30) 46) (= (travel-fast n15 n35) 61) (=
(travel-fast n15 n40) 76)

(= (travel-fast n20 n25) 16) (= (travel-fast n20 n30) 31) (=
(travel-fast n20 n35) 46) (= (travel-fast n20 n40) 61)
```

```
(= (travel-fast n25 n30) 16) (= (travel-fast n25 n35) 31) (=
(travel-fast n25 n40) 46)

(= (travel-fast n30 n35) 16) (= (travel-fast n30 n40) 31)

(= (travel-fast n35 n40) 16)

(= (total-cost) 0)

)

(:goal
(and
(passenger-at p0 n1)
(passenger-at p1 n13)
(passenger-at p2 n9)
(passenger-at p3 n18)
(passenger-at p4 n25)
(passenger-at p5 n1)
(passenger-at p6 n6)
(passenger-at p7 n31)
(passenger-at p8 n40)
(passenger-at p9 n9)
(passenger-at p10 n0)
(passenger-at p11 n30)
(passenger-at p12 n39)
(passenger-at p13 n10)
(passenger-at p14 n10)
(passenger-at p15 n14)
(passenger-at p16 n32)
(passenger-at p17 n34)
(passenger-at p18 n5)
(passenger-at p19 n13)
(passenger-at p20 n20)
(passenger-at p21 n15)
(passenger-at p22 n1)
(passenger-at p23 n16)
(passenger-at p24 n20)
(passenger-at p25 n21)
(passenger-at p26 n40)
(passenger-at p27 n37)
(passenger-at p28 n34)
(passenger-at p29 n35)
(passenger-at p30 n24)
(passenger-at p31 n24)
(passenger-at p32 n7)
(passenger-at p33 n25)
(passenger-at p34 n37)
(passenger-at p35 n21)
(passenger-at p36 n8)
```

```
(passenger-at p37 n35)
(passenger-at p38 n7)
(passenger-at p39 n31)
(passenger-at p40 n21)
(passenger-at p41 n26)
(passenger-at p42 n36)
(passenger-at p43 n4)
(passenger-at p44 n33)
(passenger-at p45 n11)
(passenger-at p46 n36)
(passenger-at p47 n32)
(passenger-at p48 n14)
(passenger-at p49 n14)
(passenger-at p50 n14)
(passenger-at p51 n28)
(passenger-at p52 n31)
(passenger-at p53 n33)
(passenger-at p54 n33)
(passenger-at p55 n25)
(passenger-at p56 n36)
(passenger-at p57 n34)
(passenger-at p58 n33)
(passenger-at p59 n21)
))

(:metric minimize (total-cost))

)
```

8.3 Elevators con modificaciones: Modif.1+Modif.2+Modif.3

8.3.1. Dominio

```
(define (domain elevators-sequencedstrips)
  (:requirements :typing :action-costs)
  (:types      elevator - object
               slow-elevator fast-elevator - elevator
               passenger - object
               count - object
         )

(:predicates
     (passenger-at ?person - passenger ?floor - count)
     (boarded ?person - passenger ?lift - elevator)
     (lift-at ?lift - elevator ?floor - count)
     (reachable-floor ?lift - elevator ?floor - count)
)

(:functions (total-cost) - number
            (travel-slow ?f1 - count ?f2 - count) - number
            (travel-fast ?f1 - count ?f2 - count) - number
            (capacity   ?lift - elevator) - number
            (occupation ?lift - elevator) - number
            (floor-number ?floor - count) - number
)

(:action move-up-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f1)(floor-number ?f2)) (reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f1 ?f2))))

(:action move-down-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f2)(floor-number ?f1)) (reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f2 ?f1))))

(:action move-up-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f1)(floor-number ?f2)) (reachable-floor ?lift ?f2) )
```

```
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f1 ?f2))))

(:action move-down-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f2)(floor-number ?f1)) (reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f2 ?f1))))

(:action board
  :parameters (?p - passenger ?lift - elevator ?f - count )
  :precondition (and  (lift-at ?lift ?f) (passenger-at ?p
?f) (< (occupation ?lift)(capacity ?lift)) )
  :effect (and (not (passenger-at ?p ?f)) (boarded ?p ?lift)
(increase (occupation ?lift) 1) ))

(:action leave
  :parameters (?p - passenger ?lift - elevator ?f - count )
  :precondition (and  (lift-at ?lift ?f) (boarded ?p ?lift)
(> (occupation ?lift) 0) )
  :effect (and (passenger-at ?p ?f) (not (boarded ?p ?lift))
(decrease (occupation ?lift) 1) ))

)
```

8.3.2. Problema 1

```
(define (problem elevators-sequencedstrips-p16_14_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16  -
count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13  - passenger
fast0 fast1  - fast-elevator
slow0-0 slow1-0 - slow-elevator
)

(:init

; se le asignan valores numéricos a los pisos

(= (floor-number n0) 0)
(= (floor-number n1) 1)
(= (floor-number n2) 2)
(= (floor-number n3) 3)
(= (floor-number n4) 4)
(= (floor-number n5) 5)
(= (floor-number n6) 6)
(= (floor-number n7) 7)
(= (floor-number n8) 8)
(= (floor-number n9) 9)
(= (floor-number n10) 10)
(= (floor-number n11) 11)
(= (floor-number n12) 12)
(= (floor-number n13) 13)
(= (floor-number n14) 14)
(= (floor-number n15) 15)
(= (floor-number n16) 16)

;se inicializa la ocupación de los ascensores y se les
asigna capacidad máxima

(lift-at fast0 n8)
(= (occupation fast0) 0)
(= (capacity fast0) 4)
(reachable-floor fast0 n0)(reachable-floor fast0
n4)(reachable-floor fast0 n8)(reachable-floor fast0
n12)(reachable-floor fast0 n16)

(lift-at fast1 n12)
(= (occupation fast1) 0)
(= (capacity fast1) 4)
```

```
(reachable-floor fast1 n0)(reachable-floor fast1
n4)(reachable-floor fast1 n8)(reachable-floor fast1
n12)(reachable-floor fast1 n16)

(lift-at slow0-0 n2)
(= (occupation slow0-0) 0)
(= (capacity slow0-0) 3)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0
n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0
n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0
n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0
n7)(reachable-floor slow0-0 n8)

(lift-at slow1-0 n12)
(= (occupation slow1-0) 0)
(= (capacity slow1-0) 3)
(reachable-floor slow1-0 n8)(reachable-floor slow1-0
n9)(reachable-floor slow1-0 n10)(reachable-floor slow1-0
n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0
n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0
n15)(reachable-floor slow1-0 n16)

(passenger-at p0 n13)
(passenger-at p1 n10)
(passenger-at p2 n13)
(passenger-at p3 n0)
(passenger-at p4 n9)
(passenger-at p5 n12)
(passenger-at p6 n8)
(passenger-at p7 n3)
(passenger-at p8 n5)
(passenger-at p9 n2)
(passenger-at p10 n4)
(passenger-at p11 n11)
(passenger-at p12 n13)
(passenger-at p13 n7)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (=
(travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-
slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow
n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n1 n2)
6) (= (travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (=
(travel-slow n1 n5) 9) (= (travel-slow n1 n6) 10) (=
(travel-slow n1 n7) 11) (= (travel-slow n1 n8) 12) (=
(travel-slow n2 n3) 6) (= (travel-slow n2 n4) 7) (= (travel-
slow n2 n5) 8) (= (travel-slow n2 n6) 9) (= (travel-slow n2
n7) 10) (= (travel-slow n2 n8) 11) (= (travel-slow n3 n4) 6)
(= (travel-slow n3 n5) 7) (= (travel-slow n3 n6) 8) (=
(travel-slow n3 n7) 9) (= (travel-slow n3 n8) 10) (=
(travel-slow n4 n5) 6) (= (travel-slow n4 n6) 7)
```

```
(= (travel-slow n4 n7) 8) (= (travel-slow n4 n8) 9) (=
(travel-slow n5 n6) 6) (= (travel-slow n5 n7) 7) (= (travel-
slow n5 n8) 8) (= (travel-slow n6 n7) 6) (= (travel-slow n6
n8) 7) (= (travel-slow n7 n8) 6)

(= (travel-slow n8 n9) 6) (= (travel-slow n8 n10) 7) (=
(travel-slow n8 n11) 8) (= (travel-slow n8 n12) 9) (=
(travel-slow n8 n13) 10) (= (travel-slow n8 n14) 11) (=
(travel-slow n8 n15) 12) (= (travel-slow n8 n16) 13) (=
(travel-slow n9 n10) 6) (= (travel-slow n9 n11) 7) (=
(travel-slow n9 n12) 8) (= (travel-slow n9 n13) 9) (=
(travel-slow n9 n14) 10) (= (travel-slow n9 n15) 11) (=
(travel-slow n9 n16) 12) (= (travel-slow n10 n11) 6) (=
(travel-slow n10 n12) 7) (= (travel-slow n10 n13) 8) (=
(travel-slow n10 n14) 9) (= (travel-slow n10 n15) 10) (=
(travel-slow n10 n16) 11) (= (travel-slow n11 n12) 6) (=
(travel-slow n11 n13) 7) (= (travel-slow n11 n14) 8) (=
(travel-slow n11 n15) 9) (= (travel-slow n11 n16) 10) (=
(travel-slow n12 n13) 6) (= (travel-slow n12 n14) 7) (=
(travel-slow n12 n15) 8) (= (travel-slow n12 n16) 9) (=
(travel-slow n13 n14) 6) (= (travel-slow n13 n15) 7) (=
(travel-slow n13 n16) 8) (= (travel-slow n14 n15) 6) (=
(travel-slow n14 n16) 7) (= (travel-slow n15 n16) 6)


(= (travel-fast n0 n4) 13) (= (travel-fast n0 n8) 25) (=
(travel-fast n0 n12) 37) (= (travel-fast n0 n16) 49)

(= (travel-fast n4 n8) 13) (= (travel-fast n4 n12) 25) (=
(travel-fast n4 n16) 37)

(= (travel-fast n8 n12) 13) (= (travel-fast n8 n16) 25)

(= (travel-fast n12 n16) 13)

(= (total-cost) 0)

)

(:goal
(and
(passenger-at p0 n8)
(passenger-at p1 n15)
(passenger-at p2 n6)
(passenger-at p3 n14)
(passenger-at p4 n5)
(passenger-at p5 n2)
(passenger-at p6 n14)
(passenger-at p7 n4)
(passenger-at p8 n10)
```

```
(passenger-at p9 n9)
(passenger-at p10 n12)
(passenger-at p11 n13)
(passenger-at p12 n5)
(passenger-at p13 n6)
))

(:metric minimize (total-cost))

)
```

8.3.3. Problema 20

```
(define (problem elevators-sequencedstrips-p40_60_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16
n17 n18 n19 n20 n21 n22 n23 n24 n25 n26 n27 n28 n29 n30 n31
n32 n33 n34 n35 n36 n37 n38 n39 n40  - count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16
p17 p18 p19 p20 p21 p22 p23 p24 p25 p26 p27 p28 p29 p30 p31
p32 p33 p34 p35 p36 p37 p38 p39 p40 p41 p42 p43 p44 p45 p46
p47 p48 p49 p50 p51 p52 p53 p54 p55 p56 p57 p58 p59  -
passenger
fast0 fast1 fast2 fast3  - fast-elevator
slow0-0 slow1-0 slow2-0 slow3-0 - slow-elevator
)

(:init

; se le asignan valores numéricos a los pisos

(= (floor-number n0) 0)
(= (floor-number n1) 1)
(= (floor-number n2) 2)
(= (floor-number n3) 3)
(= (floor-number n4) 4)
(= (floor-number n5) 5)
(= (floor-number n6) 6)
(= (floor-number n7) 7)
(= (floor-number n8) 8)
(= (floor-number n9) 9)
(= (floor-number n10) 10)
(= (floor-number n11) 11)
(= (floor-number n12) 12)
(= (floor-number n13) 13)
(= (floor-number n14) 14)
(= (floor-number n15) 15)
(= (floor-number n16) 16)
(= (floor-number n17) 17)
(= (floor-number n18) 18)
(= (floor-number n19) 19)
(= (floor-number n20) 20)
(= (floor-number n21) 21)
(= (floor-number n22) 22)
(= (floor-number n23) 23)
(= (floor-number n24) 24)
(= (floor-number n25) 25)
(= (floor-number n26) 26)
```

```
(= (floor-number n27) 27)
(= (floor-number n28) 28)
(= (floor-number n29) 29)
(= (floor-number n30) 30)
(= (floor-number n31) 31)
(= (floor-number n32) 32)
(= (floor-number n33) 33)
(= (floor-number n34) 34)
(= (floor-number n35) 35)
(= (floor-number n36) 36)
(= (floor-number n37) 37)
(= (floor-number n38) 38)
(= (floor-number n39) 39)
(= (floor-number n40) 40)

;se inicializa la ocupación de los ascensores y se les
asigna capacidad máxima

(lift-at fast0 n10)
(= (occupation fast0) 0)
(= (capacity fast0) 6)
(reachable-floor fast0 n0)(reachable-floor fast0
n5)(reachable-floor fast0 n10)(reachable-floor fast0
n15)(reachable-floor fast0 n20)(reachable-floor fast0
n25)(reachable-floor fast0 n30)(reachable-floor fast0
n35)(reachable-floor fast0 n40)

(lift-at fast1 n40)
(= (occupation fast1) 0)
(= (capacity fast1) 6)
(reachable-floor fast1 n0)(reachable-floor fast1
n5)(reachable-floor fast1 n10)(reachable-floor fast1
n15)(reachable-floor fast1 n20)(reachable-floor fast1
n25)(reachable-floor fast1 n30)(reachable-floor fast1
n35)(reachable-floor fast1 n40)

(lift-at fast2 n0)
(= (occupation fast2) 0)
(= (capacity fast2) 6)
(reachable-floor fast2 n0)(reachable-floor fast2
n5)(reachable-floor fast2 n10)(reachable-floor fast2
n15)(reachable-floor fast2 n20)(reachable-floor fast2
n25)(reachable-floor fast2 n30)(reachable-floor fast2
n35)(reachable-floor fast2 n40)

(lift-at fast3 n20)
(= (occupation fast3) 0)
(= (capacity fast3) 6)
(reachable-floor fast3 n0)(reachable-floor fast3
n5)(reachable-floor fast3 n10)
```

```
(reachable-floor fast3 n15)(reachable-floor fast3
n20)(reachable-floor fast3 n25)(reachable-floor fast3
n30)(reachable-floor fast3 n35)(reachable-floor fast3 n40)

(lift-at slow0-0 n9)
(= (occupation slow0-0) 0)
(= (capacity slow0-0) 4)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0
n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0
n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0
n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0
n7)(reachable-floor slow0-0 n8)(reachable-floor slow0-0
n9)(reachable-floor slow0-0 n10)

(lift-at slow1-0 n12)
(= (occupation slow1-0) 0)
(= (capacity slow1-0) 4)
(reachable-floor slow1-0 n10)(reachable-floor slow1-0
n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0
n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0
n15)(reachable-floor slow1-0 n16)(reachable-floor slow1-0
n17)(reachable-floor slow1-0 n18)(reachable-floor slow1-0
n19)(reachable-floor slow1-0 n20)

(lift-at slow2-0 n23)
(= (occupation slow2-0) 0)
(= (capacity slow2-0) 4)
(reachable-floor slow2-0 n20)(reachable-floor slow2-0
n21)(reachable-floor slow2-0 n22)(reachable-floor slow2-0
n23)(reachable-floor slow2-0 n24)(reachable-floor slow2-0
n25)(reachable-floor slow2-0 n26)(reachable-floor slow2-0
n27)(reachable-floor slow2-0 n28)(reachable-floor slow2-0
n29)(reachable-floor slow2-0 n30)

(lift-at slow3-0 n36)
(= (occupation slow3-0) 0)
(= (capacity slow3-0) 4)
(reachable-floor slow3-0 n30)(reachable-floor slow3-0
n31)(reachable-floor slow3-0 n32)(reachable-floor slow3-0
n33)(reachable-floor slow3-0 n34)(reachable-floor slow3-0
n35)(reachable-floor slow3-0 n36)(reachable-floor slow3-0
n37)(reachable-floor slow3-0 n38)(reachable-floor slow3-0
n39)(reachable-floor slow3-0 n40)

(passenger-at p0 n18)
(passenger-at p1 n32)
(passenger-at p2 n17)
(passenger-at p3 n32)
(passenger-at p4 n35)
(passenger-at p5 n3)
```

```
(passenger-at p6 n38)
(passenger-at p7 n38)
(passenger-at p8 n36)
(passenger-at p9 n19)
(passenger-at p10 n34)
(passenger-at p11 n19)
(passenger-at p12 n2)
(passenger-at p13 n25)
(passenger-at p14 n13)
(passenger-at p15 n29)
(passenger-at p16 n1)
(passenger-at p17 n11)
(passenger-at p18 n9)
(passenger-at p19 n19)
(passenger-at p20 n8)
(passenger-at p21 n11)
(passenger-at p22 n17)
(passenger-at p23 n11)
(passenger-at p24 n21)
(passenger-at p25 n10)
(passenger-at p26 n39)
(passenger-at p27 n24)
(passenger-at p28 n25)
(passenger-at p29 n9)
(passenger-at p30 n22)
(passenger-at p31 n37)
(passenger-at p32 n15)
(passenger-at p33 n18)
(passenger-at p34 n12)
(passenger-at p35 n38)
(passenger-at p36 n17)
(passenger-at p37 n36)
(passenger-at p38 n10)
(passenger-at p39 n10)
(passenger-at p40 n27)
(passenger-at p41 n12)
(passenger-at p42 n20)
(passenger-at p43 n23)
(passenger-at p44 n29)
(passenger-at p45 n40)
(passenger-at p46 n16)
(passenger-at p47 n36)
(passenger-at p48 n2)
(passenger-at p49 n16)
(passenger-at p50 n11)
(passenger-at p51 n35)
(passenger-at p52 n23)
(passenger-at p53 n22)
(passenger-at p54 n38)
(passenger-at p55 n24)
```

```
(passenger-at p56 n13)
(passenger-at p57 n10)
(passenger-at p58 n32)
(passenger-at p59 n38)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (=
(travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-
slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow
n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n0 n9)
14) (= (travel-slow n0 n10) 15) (= (travel-slow n1 n2) 6) (=
(travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (= (travel-
slow n1 n5) 9) (= (travel-slow n1 n6) 10) (= (travel-slow n1
n7) 11) (= (travel-slow n1 n8) 12) (= (travel-slow n1 n9)
13) (= (travel-slow n1 n10) 14) (= (travel-slow n2 n3) 6) (=
(travel-slow n2 n4) 7) (= (travel-slow n2 n5) 8) (= (travel-
slow n2 n6) 9) (= (travel-slow n2 n7) 10) (= (travel-slow n2
n8) 11) (= (travel-slow n2 n9) 12) (= (travel-slow n2 n10)
13) (= (travel-slow n3 n4) 6) (= (travel-slow n3 n5) 7) (=
(travel-slow n3 n6) 8) (= (travel-slow n3 n7) 9) (= (travel-
slow n3 n8) 10) (= (travel-slow n3 n9) 11) (= (travel-slow
n3 n10) 12) (= (travel-slow n4 n5) 6) (= (travel-slow n4 n6)
7) (= (travel-slow n4 n7) 8) (= (travel-slow n4 n8) 9) (=
(travel-slow n4 n9) 10) (= (travel-slow n4 n10) 11) (=
(travel-slow n5 n6) 6) (= (travel-slow n5 n7) 7) (= (travel-
slow n5 n8) 8) (= (travel-slow n5 n9) 9) (= (travel-slow n5
n10) 10) (= (travel-slow n6 n7) 6) (= (travel-slow n6 n8) 7)
(= (travel-slow n6 n9) 8) (= (travel-slow n6 n10) 9) (=
(travel-slow n7 n8) 6) (= (travel-slow n7 n9) 7) (= (travel-
slow n7 n10) 8) (= (travel-slow n8 n9) 6) (= (travel-slow n8
n10) 7) (= (travel-slow n9 n10) 6)

(= (travel-slow n10 n11) 6) (= (travel-slow n10 n12) 7) (=
(travel-slow n10 n13) 8) (= (travel-slow n10 n14) 9) (=
(travel-slow n10 n15) 10) (= (travel-slow n10 n16) 11) (=
(travel-slow n10 n17) 12) (= (travel-slow n10 n18) 13) (=
(travel-slow n10 n19) 14) (= (travel-slow n10 n20) 15) (=
(travel-slow n11 n12) 6) (= (travel-slow n11 n13) 7) (=
(travel-slow n11 n14) 8) (= (travel-slow n11 n15) 9) (=
(travel-slow n11 n16) 10) (= (travel-slow n11 n17) 11) (=
(travel-slow n11 n18) 12) (= (travel-slow n11 n19) 13) (=
(travel-slow n11 n20) 14) (= (travel-slow n12 n13) 6) (=
(travel-slow n12 n14) 7) (= (travel-slow n12 n15) 8) (=
(travel-slow n12 n16) 9) (= (travel-slow n12 n17) 10) (=
(travel-slow n12 n18) 11) (= (travel-slow n12 n19) 12) (=
(travel-slow n12 n20) 13) (= (travel-slow n13 n14) 6) (=
(travel-slow n13 n15) 7) (= (travel-slow n13 n16) 8) (=
(travel-slow n13 n17) 9) (= (travel-slow n13 n18) 10) (=
(travel-slow n13 n19) 11) (= (travel-slow n13 n20) 12) (=
(travel-slow n14 n15) 6) (= (travel-slow n14 n16) 7) (=
(travel-slow n14 n17) 8) (= (travel-slow n14 n18) 9) (=
```

```
(travel-slow n14 n19) 10) (= (travel-slow n14 n20) 11) (=
(travel-slow n15 n16) 6) (= (travel-slow n15 n17) 7) (=
(travel-slow n15 n18) 8) (= (travel-slow n15 n19) 9) (=
(travel-slow n15 n20) 10) (= (travel-slow n16 n17) 6) (=
(travel-slow n16 n18) 7) (= (travel-slow n16 n19) 8) (=
(travel-slow n16 n20) 9) (= (travel-slow n17 n18) 6) (=
(travel-slow n17 n19) 7) (= (travel-slow n17 n20) 8) (=
(travel-slow n18 n19) 6) (= (travel-slow n18 n20) 7) (=
(travel-slow n19 n20) 6)

(= (travel-slow n20 n21) 6) (= (travel-slow n20 n22) 7) (=
(travel-slow n20 n23) 8) (= (travel-slow n20 n24) 9) (=
(travel-slow n20 n25) 10) (= (travel-slow n20 n26) 11) (=
(travel-slow n20 n27) 12) (= (travel-slow n20 n28) 13) (=
(travel-slow n20 n29) 14) (= (travel-slow n20 n30) 15) (=
(travel-slow n21 n22) 6) (= (travel-slow n21 n23) 7) (=
(travel-slow n21 n24) 8) (= (travel-slow n21 n25) 9) (=
(travel-slow n21 n26) 10) (= (travel-slow n21 n27) 11) (=
(travel-slow n21 n28) 12) (= (travel-slow n21 n29) 13) (=
(travel-slow n21 n30) 14) (= (travel-slow n22 n23) 6) (=
(travel-slow n22 n24) 7) (= (travel-slow n22 n25) 8) (=
(travel-slow n22 n26) 9) (= (travel-slow n22 n27) 10) (=
(travel-slow n22 n28) 11) (= (travel-slow n22 n29) 12) (=
(travel-slow n22 n30) 13) (= (travel-slow n23 n24) 6) (=
(travel-slow n23 n25) 7) (= (travel-slow n23 n26) 8) (=
(travel-slow n23 n27) 9) (= (travel-slow n23 n28) 10) (=
(travel-slow n23 n29) 11) (= (travel-slow n23 n30) 12) (=
(travel-slow n24 n25) 6) (= (travel-slow n24 n26) 7) (=
(travel-slow n24 n27) 8) (= (travel-slow n24 n28) 9) (=
(travel-slow n24 n29) 10) (= (travel-slow n24 n30) 11) (=
(travel-slow n25 n26) 6) (= (travel-slow n25 n27) 7) (=
(travel-slow n25 n28) 8) (= (travel-slow n25 n29) 9) (=
(travel-slow n25 n30) 10) (= (travel-slow n26 n27) 6) (=
(travel-slow n26 n28) 7) (= (travel-slow n26 n29) 8) (=
(travel-slow n26 n30) 9) (= (travel-slow n27 n28) 6) (=
(travel-slow n27 n29) 7) (= (travel-slow n27 n30) 8) (=
(travel-slow n28 n29) 6) (= (travel-slow n28 n30) 7) (=
(travel-slow n29 n30) 6)

(= (travel-slow n30 n31) 6) (= (travel-slow n30 n32) 7) (=
(travel-slow n30 n33) 8) (= (travel-slow n30 n34) 9) (=
(travel-slow n30 n35) 10) (= (travel-slow n30 n36) 11) (=
(travel-slow n30 n37) 12) (= (travel-slow n30 n38) 13) (=
(travel-slow n30 n39) 14) (= (travel-slow n30 n40) 15) (=
(travel-slow n31 n32) 6) (= (travel-slow n31 n33) 7) (=
(travel-slow n31 n34) 8) (= (travel-slow n31 n35) 9) (=
(travel-slow n31 n36) 10) (= (travel-slow n31 n37) 11) (=
(travel-slow n31 n38) 12) (= (travel-slow n31 n39) 13) (=
(travel-slow n31 n40) 14) (= (travel-slow n32 n33) 6) (=
(travel-slow n32 n34) 7) (= (travel-slow n32 n35) 8) (=
```

```
(travel-slow n32 n36) 9) (= (travel-slow n32 n37) 10) (=
(travel-slow n32 n38) 11) (= (travel-slow n32 n39) 12) (=
(travel-slow n32 n40) 13) (= (travel-slow n33 n34) 6) (=
(travel-slow n33 n35) 7) (= (travel-slow n33 n36) 8) (=
(travel-slow n33 n37) 9) (= (travel-slow n33 n38) 10) (=
(travel-slow n33 n39) 11) (= (travel-slow n33 n40) 12) (=
(travel-slow n34 n35) 6) (= (travel-slow n34 n36) 7) (=
(travel-slow n34 n37) 8) (= (travel-slow n34 n38) 9) (=
(travel-slow n34 n39) 10) (= (travel-slow n34 n40) 11) (=
(travel-slow n35 n36) 6) (= (travel-slow n35 n37) 7) (=
(travel-slow n35 n38) 8) (= (travel-slow n35 n39) 9) (=
(travel-slow n35 n40) 10) (= (travel-slow n36 n37) 6) (=
(travel-slow n36 n38) 7) (= (travel-slow n36 n39) 8) (=
(travel-slow n36 n40) 9) (= (travel-slow n37 n38) 6) (=
(travel-slow n37 n39) 7) (= (travel-slow n37 n40) 8) (=
(travel-slow n38 n39) 6) (= (travel-slow n38 n40) 7) (=
(travel-slow n39 n40) 6)


(= (travel-fast n0 n5) 16) (= (travel-fast n0 n10) 31) (=
(travel-fast n0 n15) 46) (= (travel-fast n0 n20) 61) (=
(travel-fast n0 n25) 76) (= (travel-fast n0 n30) 91) (=
(travel-fast n0 n35) 106) (= (travel-fast n0 n40) 121)

(= (travel-fast n5 n10) 16) (= (travel-fast n5 n15) 31) (=
(travel-fast n5 n20) 46) (= (travel-fast n5 n25) 61) (=
(travel-fast n5 n30) 76) (= (travel-fast n5 n35) 91) (=
(travel-fast n5 n40) 106)

(= (travel-fast n10 n15) 16) (= (travel-fast n10 n20) 31) (=
(travel-fast n10 n25) 46) (= (travel-fast n10 n30) 61) (=
(travel-fast n10 n35) 76) (= (travel-fast n10 n40) 91)

(= (travel-fast n15 n20) 16) (= (travel-fast n15 n25) 31) (=
(travel-fast n15 n30) 46) (= (travel-fast n15 n35) 61) (=
(travel-fast n15 n40) 76)

(= (travel-fast n20 n25) 16) (= (travel-fast n20 n30) 31) (=
(travel-fast n20 n35) 46) (= (travel-fast n20 n40) 61)

(= (travel-fast n25 n30) 16) (= (travel-fast n25 n35) 31) (=
(travel-fast n25 n40) 46)

(= (travel-fast n30 n35) 16) (= (travel-fast n30 n40) 31)

(= (travel-fast n35 n40) 16)

(= (total-cost) 0)

)
```

```
(:goal
(and
(passenger-at p0 n1)
(passenger-at p1 n13)
(passenger-at p2 n9)
(passenger-at p3 n18)
(passenger-at p4 n25)
(passenger-at p5 n1)
(passenger-at p6 n6)
(passenger-at p7 n31)
(passenger-at p8 n40)
(passenger-at p9 n9)
(passenger-at p10 n0)
(passenger-at p11 n30)
(passenger-at p12 n39)
(passenger-at p13 n10)
(passenger-at p14 n10)
(passenger-at p15 n14)
(passenger-at p16 n32)
(passenger-at p17 n34)
(passenger-at p18 n5)
(passenger-at p19 n13)
(passenger-at p20 n20)
(passenger-at p21 n15)
(passenger-at p22 n1)
(passenger-at p23 n16)
(passenger-at p24 n20)
(passenger-at p25 n21)
(passenger-at p26 n40)
(passenger-at p27 n37)
(passenger-at p28 n34)
(passenger-at p29 n35)
(passenger-at p30 n24)
(passenger-at p31 n24)
(passenger-at p32 n7)
(passenger-at p33 n25)
(passenger-at p34 n37)
(passenger-at p35 n21)
(passenger-at p36 n8)
(passenger-at p37 n35)
(passenger-at p38 n7)
(passenger-at p39 n31)
(passenger-at p40 n21)
(passenger-at p41 n26)
(passenger-at p42 n36)
(passenger-at p43 n4)
(passenger-at p44 n33)
(passenger-at p45 n11)
(passenger-at p46 n36)
(passenger-at p47 n32)
```

```
(passenger-at p48 n14)
(passenger-at p49 n14)
(passenger-at p50 n14)
(passenger-at p51 n28)
(passenger-at p52 n31)
(passenger-at p53 n33)
(passenger-at p54 n33)
(passenger-at p55 n25)
(passenger-at p56 n36)
(passenger-at p57 n34)
(passenger-at p58 n33)
(passenger-at p59 n21)
))

(:metric minimize (total-cost))

)
```

8.4 Elevators con modificaciones: Modif.1+Modif.2

8.4.1. Dominio

```
(define (domain elevators-sequencedstrips)
  (:requirements :typing :action-costs)
  (:types      elevator - object
               slow-elevator fast-elevator - elevator
               passenger - object
               count - object
         )

(:predicates
     (passenger-at ?person - passenger ?floor - count)
     (boarded ?person - passenger ?lift - elevator)
     (lift-at ?lift - elevator ?floor - count)
     (reachable-floor ?lift - elevator ?floor - count)
     (above ?floor1 - count ?floor2 - count)
)

(:functions (total-cost) - number
            (travel-slow ?f1 - count ?f2 - count) - number
            (travel-fast ?f1 - count ?f2 - count) - number
            (capacity   ?lift - elevator) - number
            (occupation ?lift - elevator) - number
)

(:action move-up-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (above ?f1 ?f2 )
(reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f1 ?f2))))

(:action move-down-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (above ?f2 ?f1 )
(reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f2 ?f1))))

(:action move-up-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (above ?f1 ?f2 )
(reachable-floor ?lift ?f2) )
```

```
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f1 ?f2))))

(:action move-down-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (above ?f2 ?f1 )
(reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f2 ?f1))))

(:action board
  :parameters (?p - passenger ?lift - elevator ?f - count )
  :precondition (and  (lift-at ?lift ?f) (passenger-at ?p
?f) (< (occupation ?lift)(capacity ?lift)) )
  :effect (and (not (passenger-at ?p ?f)) (boarded ?p ?lift)
(increase (occupation ?lift) 1) ))

(:action leave
  :parameters (?p - passenger ?lift - elevator ?f - count )
  :precondition (and  (lift-at ?lift ?f) (boarded ?p ?lift)
(> (occupation ?lift) 0) )
  :effect (and (passenger-at ?p ?f) (not (boarded ?p ?lift))
(decrease (occupation ?lift) 1) ))

)
```

8.4.2. Problema 1

```
(define (problem elevators-sequencedstrips-p16_14_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16  -
count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13  - passenger
fast0 fast1  - fast-elevator
slow0-0 slow1-0 - slow-elevator
)

(:init

(above n0 n1) (above n0 n2) (above n0 n3) (above n0 n4)
(above n0 n5) (above n0 n6) (above n0 n7) (above n0 n8)
(above n0 n9) (above n0 n10) (above n0 n11) (above n0 n12)
(above n0 n13) (above n0 n14) (above n0 n15) (above n0 n16)
(above n1 n2) (above n1 n3) (above n1 n4) (above n1 n5)
(above n1 n6) (above n1 n7) (above n1 n8) (above n1 n9)
(above n1 n10) (above n1 n11) (above n1 n12) (above n1 n13)
(above n1 n14) (above n1 n15) (above n1 n16)
(above n2 n3) (above n2 n4) (above n2 n5) (above n2 n6)
(above n2 n7) (above n2 n8) (above n2 n9) (above n2 n10)
(above n2 n11) (above n2 n12) (above n2 n13) (above n2 n14)
(above n2 n15) (above n2 n16)
(above n3 n4) (above n3 n5) (above n3 n6) (above n3 n7)
(above n3 n8) (above n3 n9) (above n3 n10) (above n3 n11)
(above n3 n12) (above n3 n13) (above n3 n14) (above n3 n15)
(above n3 n16)
(above n4 n5) (above n4 n6) (above n4 n7) (above n4 n8)
(above n4 n9) (above n4 n10) (above n4 n11) (above n4 n12)
(above n4 n13) (above n4 n14) (above n4 n15) (above n4 n16)
(above n5 n6) (above n5 n7) (above n5 n8) (above n5 n9)
(above n5 n10) (above n5 n11) (above n5 n12) (above n5 n13)
(above n5 n14) (above n5 n15) (above n5 n16)
(above n6 n7) (above n6 n8) (above n6 n9) (above n6 n10)
(above n6 n11) (above n6 n12) (above n6 n13) (above n6 n14)
(above n6 n15) (above n6 n16)
(above n7 n8) (above n7 n9) (above n7 n10) (above n7 n11)
(above n7 n12) (above n7 n13) (above n7 n14) (above n7 n15)
(above n7 n16)
(above n8 n9) (above n8 n10) (above n8 n11) (above n8 n12)
(above n8 n13) (above n8 n14) (above n8 n15) (above n8 n16)
(above n9 n10) (above n9 n11) (above n9 n12) (above n9 n13)
(above n9 n14) (above n9 n15) (above n9 n16)
(above n10 n11) (above n10 n12) (above n10 n13) (above n10
n14) (above n10 n15) (above n10 n16)
```

```
(above n11 n12) (above n11 n13) (above n11 n14) (above n11
n15) (above n11 n16)
(above n12 n13) (above n12 n14) (above n12 n15) (above n12
n16)
(above n13 n14) (above n13 n15) (above n13 n16)
(above n14 n15) (above n14 n16)
(above n15 n16)

(lift-at fast0 n8)
(= (occupation fast0) 0)
(= (capacity fast0) 4)
(reachable-floor fast0 n0)(reachable-floor fast0
n4)(reachable-floor fast0 n8)(reachable-floor fast0
n12)(reachable-floor fast0 n16)

(lift-at fast1 n12)
(= (occupation fast1) 0)
(= (capacity fast1) 4)
(reachable-floor fast1 n0)(reachable-floor fast1
n4)(reachable-floor fast1 n8)(reachable-floor fast1
n12)(reachable-floor fast1 n16)

(lift-at slow0-0 n2)
(= (occupation slow0-0) 0)
(= (capacity slow0-0) 3)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0
n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0
n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0
n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0
n7)(reachable-floor slow0-0 n8)

(lift-at slow1-0 n12)
(= (occupation slow1-0) 0)
(= (capacity slow1-0) 3)
(reachable-floor slow1-0 n8)(reachable-floor slow1-0
n9)(reachable-floor slow1-0 n10)(reachable-floor slow1-0
n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0
n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0
n15)(reachable-floor slow1-0 n16)

(passenger-at p0 n13)
(passenger-at p1 n10)
(passenger-at p2 n13)
(passenger-at p3 n0)
(passenger-at p4 n9)
(passenger-at p5 n12)
(passenger-at p6 n8)
(passenger-at p7 n3)
(passenger-at p8 n5)
(passenger-at p9 n2)
```

```
(passenger-at p10 n4)
(passenger-at p11 n11)
(passenger-at p12 n13)
(passenger-at p13 n7)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (=
(travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-
slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow
n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n1 n2)
6) (= (travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (=
(travel-slow n1 n5) 9) (= (travel-slow n1 n6) 10) (=
(travel-slow n1 n7) 11) (= (travel-slow n1 n8) 12) (=
(travel-slow n2 n3) 6) (= (travel-slow n2 n4) 7) (= (travel-
slow n2 n5) 8) (= (travel-slow n2 n6) 9) (= (travel-slow n2
n7) 10) (= (travel-slow n2 n8) 11) (= (travel-slow n3 n4) 6)
(= (travel-slow n3 n5) 7) (= (travel-slow n3 n6) 8) (=
(travel-slow n3 n7) 9) (= (travel-slow n3 n8) 10) (=
(travel-slow n4 n5) 6) (= (travel-slow n4 n6) 7) (= (travel-
slow n4 n7) 8) (= (travel-slow n4 n8) 9) (= (travel-slow n5
n6) 6) (= (travel-slow n5 n7) 7) (= (travel-slow n5 n8) 8)
(= (travel-slow n6 n7) 6) (= (travel-slow n6 n8) 7) (=
(travel-slow n7 n8) 6)

(= (travel-slow n8 n9) 6) (= (travel-slow n8 n10) 7) (=
(travel-slow n8 n11) 8) (= (travel-slow n8 n12) 9) (=
(travel-slow n8 n13) 10) (= (travel-slow n8 n14) 11) (=
(travel-slow n8 n15) 12) (= (travel-slow n8 n16) 13) (=
(travel-slow n9 n10) 6) (= (travel-slow n9 n11) 7) (=
(travel-slow n9 n12) 8) (= (travel-slow n9 n13) 9) (=
(travel-slow n9 n14) 10) (= (travel-slow n9 n15) 11) (=
(travel-slow n9 n16) 12) (= (travel-slow n10 n11) 6) (=
(travel-slow n10 n12) 7) (= (travel-slow n10 n13) 8) (=
(travel-slow n10 n14) 9) (= (travel-slow n10 n15) 10) (=
(travel-slow n10 n16) 11) (= (travel-slow n11 n12) 6) (=
(travel-slow n11 n13) 7) (= (travel-slow n11 n14) 8) (=
(travel-slow n11 n15) 9) (= (travel-slow n11 n16) 10) (=
(travel-slow n12 n13) 6) (= (travel-slow n12 n14) 7) (=
(travel-slow n12 n15) 8) (= (travel-slow n12 n16) 9) (=
(travel-slow n13 n14) 6) (= (travel-slow n13 n15) 7) (=
(travel-slow n13 n16) 8) (= (travel-slow n14 n15) 6) (=
(travel-slow n14 n16) 7) (= (travel-slow n15 n16) 6)


(= (travel-fast n0 n4) 13) (= (travel-fast n0 n8) 25) (=
(travel-fast n0 n12) 37) (= (travel-fast n0 n16) 49)

(= (travel-fast n4 n8) 13) (= (travel-fast n4 n12) 25) (=
(travel-fast n4 n16) 37)

(= (travel-fast n8 n12) 13) (= (travel-fast n8 n16) 25)
```

```
(= (travel-fast n12 n16) 13)

(= (total-cost) 0)

)

(:goal
(and
(passenger-at p0 n8)
(passenger-at p1 n15)
(passenger-at p2 n6)
(passenger-at p3 n14)
(passenger-at p4 n5)
(passenger-at p5 n2)
(passenger-at p6 n14)
(passenger-at p7 n4)
(passenger-at p8 n10)
(passenger-at p9 n9)
(passenger-at p10 n12)
(passenger-at p11 n13)
(passenger-at p12 n5)
(passenger-at p13 n6)
))

(:metric minimize (total-cost))

)
```

8.4.3. Problema 20

```
(define (problem elevators-sequencedstrips-p40_60_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16
n17 n18 n19 n20 n21 n22 n23 n24 n25 n26 n27 n28 n29 n30 n31
n32 n33 n34 n35 n36 n37 n38 n39 n40  - count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16
p17 p18 p19 p20 p21 p22 p23 p24 p25 p26 p27 p28 p29 p30 p31
p32 p33 p34 p35 p36 p37 p38 p39 p40 p41 p42 p43 p44 p45 p46
p47 p48 p49 p50 p51 p52 p53 p54 p55 p56 p57 p58 p59  -
passenger
fast0 fast1 fast2 fast3  - fast-elevator
slow0-0 slow1-0 slow2-0 slow3-0 - slow-elevator
)

(:init

(above n0 n1) (above n0 n2) (above n0 n3) (above n0 n4)
(above n0 n5) (above n0 n6) (above n0 n7) (above n0 n8)
(above n0 n9) (above n0 n10) (above n0 n11) (above n0 n12)
(above n0 n13) (above n0 n14) (above n0 n15) (above n0 n16)
(above n0 n17) (above n0 n18) (above n0 n19) (above n0 n20)
(above n0 n21) (above n0 n22) (above n0 n23) (above n0 n24)
(above n0 n25) (above n0 n26) (above n0 n27) (above n0 n28)
(above n0 n29) (above n0 n30) (above n0 n31) (above n0 n32)
(above n0 n33) (above n0 n34) (above n0 n35) (above n0 n36)
(above n0 n37) (above n0 n38) (above n0 n39) (above n0 n40)
(above n1 n2) (above n1 n3) (above n1 n4) (above n1 n5)
(above n1 n6) (above n1 n7) (above n1 n8) (above n1 n9)
(above n1 n10) (above n1 n11) (above n1 n12) (above n1 n13)
(above n1 n14) (above n1 n15) (above n1 n16) (above n1 n17)
(above n1 n18) (above n1 n19) (above n1 n20) (above n1 n21)
(above n1 n22) (above n1 n23) (above n1 n24) (above n1 n25)
(above n1 n26) (above n1 n27) (above n1 n28) (above n1 n29)
(above n1 n30) (above n1 n31) (above n1 n32) (above n1 n33)
(above n1 n34) (above n1 n35) (above n1 n36) (above n1 n37)
(above n1 n38) (above n1 n39) (above n1 n40)
(above n2 n3) (above n2 n4) (above n2 n5) (above n2 n6)
(above n2 n7) (above n2 n8) (above n2 n9) (above n2 n10)
(above n2 n11) (above n2 n12) (above n2 n13) (above n2 n14)
(above n2 n15) (above n2 n16) (above n2 n17) (above n2 n18)
(above n2 n19) (above n2 n20) (above n2 n21) (above n2 n22)
(above n2 n23) (above n2 n24) (above n2 n25) (above n2 n26)
(above n2 n27) (above n2 n28) (above n2 n29) (above n2 n30)
(above n2 n31) (above n2 n32) (above n2 n33) (above n2 n34)
```

```
(above n2 n35) (above n2 n36) (above n2 n37) (above n2 n38)
(above n2 n39) (above n2 n40)
(above n3 n4) (above n3 n5) (above n3 n6) (above n3 n7)
(above n3 n8) (above n3 n9) (above n3 n10) (above n3 n11)
(above n3 n12) (above n3 n13) (above n3 n14) (above n3 n15)
(above n3 n16) (above n3 n17) (above n3 n18) (above n3 n19)
(above n3 n20) (above n3 n21) (above n3 n22) (above n3 n23)
(above n3 n24) (above n3 n25) (above n3 n26) (above n3 n27)
(above n3 n28) (above n3 n29) (above n3 n30) (above n3 n31)
(above n3 n32) (above n3 n33) (above n3 n34) (above n3 n35)
(above n3 n36) (above n3 n37) (above n3 n38) (above n3 n39)
(above n3 n40)
(above n4 n5) (above n4 n6) (above n4 n7) (above n4 n8)
(above n4 n9) (above n4 n10) (above n4 n11) (above n4 n12)
(above n4 n13) (above n4 n14) (above n4 n15) (above n4 n16)
(above n4 n17) (above n4 n18) (above n4 n19) (above n4 n20)
(above n4 n21) (above n4 n22) (above n4 n23) (above n4 n24)
(above n4 n25) (above n4 n26) (above n4 n27) (above n4 n28)
(above n4 n29) (above n4 n30) (above n4 n31) (above n4 n32)
(above n4 n33) (above n4 n34) (above n4 n35) (above n4 n36)
(above n4 n37) (above n4 n38) (above n4 n39) (above n4 n40)
(above n5 n6) (above n5 n7) (above n5 n8) (above n5 n9)
(above n5 n10) (above n5 n11) (above n5 n12) (above n5 n13)
(above n5 n14) (above n5 n15) (above n5 n16) (above n5 n17)
(above n5 n18) (above n5 n19) (above n5 n20) (above n5 n21)
(above n5 n22) (above n5 n23) (above n5 n24) (above n5 n25)
(above n5 n26) (above n5 n27) (above n5 n28) (above n5 n29)
(above n5 n30) (above n5 n31) (above n5 n32) (above n5 n33)
(above n5 n34) (above n5 n35) (above n5 n36) (above n5 n37)
(above n5 n38) (above n5 n39) (above n5 n40)
(above n6 n7) (above n6 n8) (above n6 n9) (above n6 n10)
(above n6 n11) (above n6 n12) (above n6 n13) (above n6 n14)
(above n6 n15) (above n6 n16) (above n6 n17) (above n6 n18)
(above n6 n19) (above n6 n20) (above n6 n21) (above n6 n22)
(above n6 n23) (above n6 n24) (above n6 n25) (above n6 n26)
(above n6 n27) (above n6 n28) (above n6 n29) (above n6 n30)
(above n6 n31) (above n6 n32) (above n6 n33) (above n6 n34)
(above n6 n35) (above n6 n36) (above n6 n37) (above n6 n38)
(above n6 n39) (above n6 n40)
(above n7 n8) (above n7 n9) (above n7 n10) (above n7 n11)
(above n7 n12) (above n7 n13) (above n7 n14) (above n7 n15)
(above n7 n16) (above n7 n17) (above n7 n18) (above n7 n19)
(above n7 n20) (above n7 n21) (above n7 n22) (above n7 n23)
(above n7 n24) (above n7 n25) (above n7 n26) (above n7 n27)
(above n7 n28) (above n7 n29) (above n7 n30) (above n7 n31)
(above n7 n32) (above n7 n33) (above n7 n34) (above n7 n35)
(above n7 n36) (above n7 n37) (above n7 n38) (above n7 n39)
(above n7 n40)
```

(above n8 n9) (above n8 n10) (above n8 n11) (above n8 n12)
(above n8 n13) (above n8 n14) (above n8 n15) (above n8 n16)
(above n8 n17) (above n8 n18) (above n8 n19) (above n8 n20)

(above n8 n21) (above n8 n22) (above n8 n23) (above n8 n24)
(above n8 n25) (above n8 n26) (above n8 n27) (above n8 n28)
(above n8 n29) (above n8 n30) (above n8 n31) (above n8 n32)
(above n8 n33) (above n8 n34) (above n8 n35) (above n8 n36)
(above n8 n37) (above n8 n38) (above n8 n39) (above n8 n40)
(above n9 n10) (above n9 n11) (above n9 n12) (above n9 n13)
(above n9 n14) (above n9 n15) (above n9 n16) (above n9 n17)
(above n9 n18) (above n9 n19) (above n9 n20) (above n9 n21)
(above n9 n22) (above n9 n23) (above n9 n24) (above n9 n25)
(above n9 n26) (above n9 n27) (above n9 n28) (above n9 n29)
(above n9 n30) (above n9 n31) (above n9 n32) (above n9 n33)
(above n9 n34) (above n9 n35) (above n9 n36) (above n9 n37)
(above n9 n38) (above n9 n39) (above n9 n40)

(above n10 n11) (above n10 n12) (above n10 n13) (above n10 n14) (above n10 n15) (above n10 n16) (above n10 n17) (above n10 n18) (above n10 n19) (above n10 n20) (above n10 n21) (above n10 n22) (above n10 n23) (above n10 n24) (above n10 n25) (above n10 n26) (above n10 n27) (above n10 n28) (above n10 n29) (above n10 n30) (above n10 n31) (above n10 n32) (above n10 n33) (above n10 n34) (above n10 n35) (above n10 n36) (above n10 n37) (above n10 n38) (above n10 n39) (above n10 n40)

(above n11 n12) (above n11 n13) (above n11 n14) (above n11 n15) (above n11 n16) (above n11 n17) (above n11 n18) (above n11 n19) (above n11 n20) (above n11 n21) (above n11 n22) (above n11 n23) (above n11 n24) (above n11 n25) (above n11 n26) (above n11 n27) (above n11 n28) (above n11 n29) (above n11 n30) (above n11 n31) (above n11 n32) (above n11 n33) (above n11 n34) (above n11 n35) (above n11 n36) (above n11 n37) (above n11 n38) (above n11 n39) (above n11 n40)
(above n12 n13) (above n12 n14) (above n12 n15) (above n12 n16) (above n12 n17) (above n12 n18) (above n12 n19) (above n12 n20) (above n12 n21) (above n12 n22) (above n12 n23) (above n12 n24) (above n12 n25) (above n12 n26) (above n12 n27) (above n12 n28) (above n12 n29) (above n12 n30) (above n12 n31) (above n12 n32) (above n12 n33) (above n12 n34) (above n12 n35) (above n12 n36) (above n12 n37) (above n12 n38) (above n12 n39) (above n12 n40)

(above n13 n14) (above n13 n15) (above n13 n16) (above n13 n17) (above n13 n18) (above n13 n19) (above n13 n20) (above n13 n21) (above n13 n22) (above n13 n23) (above n13 n24) (above n13 n25) (above n13 n26) (above n13 n27) (above n13 n28) (above n13 n29) (above n13 n30) (above n13 n31)

(above n13 n32) (above n13 n33) (above n13 n34) (above n13 n35) (above n13 n36) (above n13 n37) (above n13 n38) (above n13 n39) (above n13 n40)
(above n14 n15) (above n14 n16) (above n14 n17) (above n14 n18) (above n14 n19) (above n14 n20) (above n14 n21) (above n14 n22) (above n14 n23) (above n14 n24) (above n14 n25) (above n14 n26) (above n14 n27) (above n14 n28) (above n14 n29) (above n14 n30) (above n14 n31) (above n14 n32) (above n14 n33) (above n14 n34) (above n14 n35) (above n14 n36) (above n14 n37) (above n14 n38) (above n14 n39) (above n14 n40)

(above n15 n16) (above n15 n17) (above n15 n18) (above n15 n19) (above n15 n20) (above n15 n21) (above n15 n22) (above n15 n23) (above n15 n24) (above n15 n25) (above n15 n26) (above n15 n27) (above n15 n28) (above n15 n29) (above n15 n30) (above n15 n31) (above n15 n32) (above n15 n33) (above n15 n34) (above n15 n35) (above n15 n36) (above n15 n37) (above n15 n38) (above n15 n39) (above n15 n40)
(above n16 n17) (above n16 n18) (above n16 n19) (above n16 n20) (above n16 n21) (above n16 n22) (above n16 n23) (above n16 n24) (above n16 n25) (above n16 n26) (above n16 n27) (above n16 n28) (above n16 n29) (above n16 n30) (above n16 n31) (above n16 n32) (above n16 n33) (above n16 n34) (above n16 n35) (above n16 n36) (above n16 n37) (above n16 n38) (above n16 n39) (above n16 n40)

(above n17 n18) (above n17 n19) (above n17 n20) (above n17 n21) (above n17 n22) (above n17 n23) (above n17 n24) (above n17 n25) (above n17 n26) (above n17 n27) (above n17 n28) (above n17 n29) (above n17 n30) (above n17 n31) (above n17 n32) (above n17 n33) (above n17 n34) (above n17 n35) (above n17 n36) (above n17 n37) (above n17 n38) (above n17 n39) (above n17 n40)

(above n18 n19) (above n18 n20) (above n18 n21) (above n18 n22) (above n18 n23) (above n18 n24) (above n18 n25) (above n18 n26) (above n18 n27) (above n18 n28) (above n18 n29) (above n18 n30) (above n18 n31) (above n18 n32) (above n18 n33) (above n18 n34) (above n18 n35) (above n18 n36) (above n18 n37) (above n18 n38) (above n18 n39) (above n18 n40)
(above n19 n20) (above n19 n21) (above n19 n22) (above n19 n23) (above n19 n24) (above n19 n25) (above n19 n26) (above n19 n27) (above n19 n28) (above n19 n29) (above n19 n30) (above n19 n31) (above n19 n32) (above n19 n33) (above n19 n34) (above n19 n35) (above n19 n36) (above n19 n37) (above n19 n38) (above n19 n39) (above n19 n40)

(above n20 n21) (above n20 n22) (above n20 n23) (above n20 n24) (above n20 n25) (above n20 n26) (above n20 n27)

(above n20 n28) (above n20 n29) (above n20 n30) (above n20 n31) (above n20 n32) (above n20 n33) (above n20 n34) (above n20 n35) (above n20 n36) (above n20 n37) (above n20 n38) (above n20 n39) (above n20 n40)
(above n21 n22) (above n21 n23) (above n21 n24) (above n21 n25) (above n21 n26) (above n21 n27) (above n21 n28) (above n21 n29) (above n21 n30) (above n21 n31) (above n21 n32) (above n21 n33) (above n21 n34) (above n21 n35) (above n21 n36) (above n21 n37) (above n21 n38) (above n21 n39) (above n21 n40)
(above n22 n23) (above n22 n24) (above n22 n25) (above n22 n26) (above n22 n27) (above n22 n28) (above n22 n29)

(above n22 n30) (above n22 n31) (above n22 n32) (above n22 n33) (above n22 n34) (above n22 n35) (above n22 n36) (above n22 n37) (above n22 n38) (above n22 n39) (above n22 n40)
(above n23 n24) (above n23 n25) (above n23 n26) (above n23 n27) (above n23 n28) (above n23 n29) (above n23 n30) (above n23 n31) (above n23 n32) (above n23 n33) (above n23 n34) (above n23 n35) (above n23 n36) (above n23 n37) (above n23 n38) (above n23 n39) (above n23 n40)
(above n24 n25) (above n24 n26) (above n24 n27) (above n24 n28) (above n24 n29) (above n24 n30) (above n24 n31) (above n24 n32) (above n24 n33) (above n24 n34) (above n24 n35) (above n24 n36) (above n24 n37) (above n24 n38) (above n24 n39) (above n24 n40)
(above n25 n26) (above n25 n27) (above n25 n28) (above n25 n29) (above n25 n30) (above n25 n31) (above n25 n32) (above n25 n33) (above n25 n34) (above n25 n35) (above n25 n36) (above n25 n37) (above n25 n38) (above n25 n39) (above n25 n40)

(above n26 n27) (above n26 n28) (above n26 n29) (above n26 n30) (above n26 n31) (above n26 n32) (above n26 n33) (above n26 n34) (above n26 n35) (above n26 n36) (above n26 n37) (above n26 n38) (above n26 n39) (above n26 n40)
(above n27 n28) (above n27 n29) (above n27 n30) (above n27 n31) (above n27 n32) (above n27 n33) (above n27 n34) (above n27 n35) (above n27 n36) (above n27 n37) (above n27 n38) (above n27 n39) (above n27 n40)
(above n28 n29) (above n28 n30) (above n28 n31) (above n28 n32) (above n28 n33) (above n28 n34) (above n28 n35) (above n28 n36) (above n28 n37) (above n28 n38) (above n28 n39) (above n28 n40)

(above n29 n30) (above n29 n31) (above n29 n32) (above n29 n33) (above n29 n34) (above n29 n35) (above n29 n36) (above n29 n37) (above n29 n38) (above n29 n39) (above n29 n40)

```
(above n30 n31) (above n30 n32) (above n30 n33) (above n30
n34) (above n30 n35) (above n30 n36) (above n30 n37) (above
n30 n38) (above n30 n39) (above n30 n40)
(above n31 n32) (above n31 n33) (above n31 n34) (above n31
n35) (above n31 n36) (above n31 n37) (above n31 n38) (above
n31 n39) (above n31 n40)
(above n32 n33) (above n32 n34) (above n32 n35) (above n32
n36) (above n32 n37) (above n32 n38) (above n32 n39) (above
n32 n40)
(above n33 n34) (above n33 n35) (above n33 n36) (above n33
n37) (above n33 n38) (above n33 n39) (above n33 n40)
(above n34 n35) (above n34 n36) (above n34 n37) (above n34
n38) (above n34 n39) (above n34 n40)
(above n35 n36) (above n35 n37) (above n35 n38) (above n35
n39) (above n35 n40)
(above n36 n37) (above n36 n38) (above n36 n39) (above n36
n40)

(above n37 n38) (above n37 n39) (above n37 n40)
(above n38 n39) (above n38 n40)
(above n39 n40)

(lift-at fast0 n10)
(= (occupation fast0) 0)
(= (capacity fast0) 6)
(reachable-floor fast0 n0)(reachable-floor fast0
n5)(reachable-floor fast0 n10)(reachable-floor fast0
n15)(reachable-floor fast0 n20)(reachable-floor fast0
n25)(reachable-floor fast0 n30)(reachable-floor fast0
n35)(reachable-floor fast0 n40)

(lift-at fast1 n40)
(= (occupation fast1) 0)
(= (capacity fast1) 6)
(reachable-floor fast1 n0)(reachable-floor fast1
n5)(reachable-floor fast1 n10)(reachable-floor fast1
n15)(reachable-floor fast1 n20)(reachable-floor fast1
n25)(reachable-floor fast1 n30)(reachable-floor fast1
n35)(reachable-floor fast1 n40)

(lift-at fast2 n0)
(= (occupation fast2) 0)
(= (capacity fast2) 6)
(reachable-floor fast2 n0)(reachable-floor fast2
n5)(reachable-floor fast2 n10)(reachable-floor fast2
n15)(reachable-floor fast2 n20)(reachable-floor fast2
n25)(reachable-floor fast2 n30)(reachable-floor fast2
n35)(reachable-floor fast2 n40)
```

```
(lift-at fast3 n20)
(= (occupation fast3) 0)
(= (capacity fast3) 6)
(reachable-floor fast3 n0)(reachable-floor fast3
n5)(reachable-floor fast3 n10)(reachable-floor fast3
n15)(reachable-floor fast3 n20)(reachable-floor fast3
n25)(reachable-floor fast3 n30)(reachable-floor fast3
n35)(reachable-floor fast3 n40)

(lift-at slow0-0 n9)
(= (occupation slow0-0) 0)
(= (capacity slow0-0) 4)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0
n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0
n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0
n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0
n7)(reachable-floor slow0-0 n8)(reachable-floor slow0-0
n9)(reachable-floor slow0-0 n10)


(lift-at slow1-0 n12)
(= (occupation slow1-0) 0)
(= (capacity slow1-0) 4)
(reachable-floor slow1-0 n10)(reachable-floor slow1-0
n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0
n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0
n15)(reachable-floor slow1-0 n16)(reachable-floor slow1-0
n17)(reachable-floor slow1-0 n18)(reachable-floor slow1-0
n19)(reachable-floor slow1-0 n20)

(lift-at slow2-0 n23)
(= (occupation slow2-0) 0)
(= (capacity slow2-0) 4)
(reachable-floor slow2-0 n20)(reachable-floor slow2-0
n21)(reachable-floor slow2-0 n22)(reachable-floor slow2-0
n23)(reachable-floor slow2-0 n24)(reachable-floor slow2-0
n25)(reachable-floor slow2-0 n26)(reachable-floor slow2-0
n27)(reachable-floor slow2-0 n28)(reachable-floor slow2-0
n29)(reachable-floor slow2-0 n30)

(lift-at slow3-0 n36)
(= (occupation slow3-0) 0)
(= (capacity slow3-0) 4)
(reachable-floor slow3-0 n30)(reachable-floor slow3-0
n31)(reachable-floor slow3-0 n32)(reachable-floor slow3-0
n33)(reachable-floor slow3-0 n34)(reachable-floor slow3-0
n35)(reachable-floor slow3-0 n36)(reachable-floor slow3-0
n37)(reachable-floor slow3-0 n38)(reachable-floor slow3-0
n39)(reachable-floor slow3-0 n40)
```

```
(passenger-at p0 n18)
(passenger-at p1 n32)
(passenger-at p2 n17)
(passenger-at p3 n32)
(passenger-at p4 n35)
(passenger-at p5 n3)
(passenger-at p6 n38)
(passenger-at p7 n38)
(passenger-at p8 n36)
(passenger-at p9 n19)
(passenger-at p10 n34)
(passenger-at p11 n19)
(passenger-at p12 n2)
(passenger-at p13 n25)
(passenger-at p14 n13)
(passenger-at p15 n29)
(passenger-at p16 n1)
(passenger-at p17 n11)
(passenger-at p18 n9)
(passenger-at p19 n19)
(passenger-at p20 n8)
(passenger-at p21 n11)
(passenger-at p22 n17)
(passenger-at p23 n11)
(passenger-at p24 n21)
(passenger-at p25 n10)
(passenger-at p26 n39)
(passenger-at p27 n24)
(passenger-at p28 n25)
(passenger-at p29 n9)
(passenger-at p30 n22)
(passenger-at p31 n37)
(passenger-at p32 n15)
(passenger-at p33 n18)
(passenger-at p34 n12)
(passenger-at p35 n38)
(passenger-at p36 n17)
(passenger-at p37 n36)
(passenger-at p38 n10)
(passenger-at p39 n10)
(passenger-at p40 n27)
(passenger-at p41 n12)
(passenger-at p42 n20)
(passenger-at p43 n23)
(passenger-at p44 n29)
(passenger-at p45 n40)
(passenger-at p46 n16)
(passenger-at p47 n36)
(passenger-at p48 n2)
(passenger-at p49 n16)
```

```
(passenger-at p50 n11)
(passenger-at p51 n35)
(passenger-at p52 n23)
(passenger-at p53 n22)
(passenger-at p54 n38)
(passenger-at p55 n24)
(passenger-at p56 n13)
(passenger-at p57 n10)
(passenger-at p58 n32)
(passenger-at p59 n38)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (=
(travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-
slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow
n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n0 n9)
14) (= (travel-slow n0 n10) 15) (= (travel-slow n1 n2) 6) (=
(travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (= (travel-
slow n1 n5) 9) (= (travel-slow n1 n6) 10) (= (travel-slow n1
n7) 11) (= (travel-slow n1 n8) 12) (= (travel-slow n1 n9)
13) (= (travel-slow n1 n10) 14) (= (travel-slow n2 n3) 6) (=

(travel-slow n2 n4) 7) (= (travel-slow n2 n5) 8) (= (travel-
slow n2 n6) 9) (= (travel-slow n2 n7) 10) (= (travel-slow n2
n8) 11) (= (travel-slow n2 n9) 12) (= (travel-slow n2 n10)
13) (= (travel-slow n3 n4) 6) (= (travel-slow n3 n5) 7) (=
(travel-slow n3 n6) 8) (= (travel-slow n3 n7) 9) (= (travel-
slow n3 n8) 10) (= (travel-slow n3 n9) 11) (= (travel-slow
n3 n10) 12) (= (travel-slow n4 n5) 6) (= (travel-slow n4 n6)
7) (= (travel-slow n4 n7) 8) (= (travel-slow n4 n8) 9) (=
(travel-slow n4 n9) 10) (= (travel-slow n4 n10) 11) (=
(travel-slow n5 n6) 6) (= (travel-slow n5 n7) 7) (= (travel-
slow n5 n8) 8) (= (travel-slow n5 n9) 9) (= (travel-slow n5
n10) 10) (= (travel-slow n6 n7) 6) (= (travel-slow n6 n8) 7)
(= (travel-slow n6 n9) 8) (= (travel-slow n6 n10) 9) (=
(travel-slow n7 n8) 6) (= (travel-slow n7 n9) 7) (= (travel-
slow n7 n10) 8) (= (travel-slow n8 n9) 6) (= (travel-slow n8
n10) 7) (= (travel-slow n9 n10) 6)

(= (travel-slow n10 n11) 6) (= (travel-slow n10 n12) 7) (=
(travel-slow n10 n13) 8) (= (travel-slow n10 n14) 9) (=
(travel-slow n10 n15) 10) (= (travel-slow n10 n16) 11) (=
(travel-slow n10 n17) 12) (= (travel-slow n10 n18) 13) (=
(travel-slow n10 n19) 14) (= (travel-slow n10 n20) 15) (=
(travel-slow n11 n12) 6) (= (travel-slow n11 n13) 7) (=
(travel-slow n11 n14) 8) (= (travel-slow n11 n15) 9) (=
(travel-slow n11 n16) 10) (= (travel-slow n11 n17) 11) (=
(travel-slow n11 n18) 12) (= (travel-slow n11 n19) 13) (=
(travel-slow n11 n20) 14) (= (travel-slow n12 n13) 6) (=
(travel-slow n12 n14) 7) (= (travel-slow n12 n15) 8) (=
(travel-slow n12 n16) 9) (= (travel-slow n12 n17) 10) (=
```

```
(travel-slow n12 n18) 11) (= (travel-slow n12 n19) 12) (=
(travel-slow n12 n20) 13) (= (travel-slow n13 n14) 6) (=
(travel-slow n13 n15) 7) (= (travel-slow n13 n16) 8) (=
(travel-slow n13 n17) 9) (= (travel-slow n13 n18) 10) (=
(travel-slow n13 n19) 11) (= (travel-slow n13 n20) 12) (=
(travel-slow n14 n15) 6) (= (travel-slow n14 n16) 7) (=
(travel-slow n14 n17) 8) (= (travel-slow n14 n18) 9) (=
(travel-slow n14 n19) 10) (= (travel-slow n14 n20) 11) (=
(travel-slow n15 n16) 6) (= (travel-slow n15 n17) 7) (=
(travel-slow n15 n18) 8) (= (travel-slow n15 n19) 9) (=
(travel-slow n15 n20) 10) (= (travel-slow n16 n17) 6) (=
(travel-slow n16 n18) 7) (= (travel-slow n16 n19) 8) (=
(travel-slow n16 n20) 9) (= (travel-slow n17 n18) 6) (=
(travel-slow n17 n19) 7) (= (travel-slow n17 n20) 8) (=
(travel-slow n18 n19) 6) (= (travel-slow n18 n20) 7) (=
(travel-slow n19 n20) 6)

(= (travel-slow n20 n21) 6) (= (travel-slow n20 n22) 7) (=
(travel-slow n20 n23) 8) (= (travel-slow n20 n24) 9) (=
(travel-slow n20 n25) 10) (= (travel-slow n20 n26) 11) (=
(travel-slow n20 n27) 12) (= (travel-slow n20 n28) 13) (=

(travel-slow n20 n29) 14) (= (travel-slow n20 n30) 15) (=
(travel-slow n21 n22) 6) (= (travel-slow n21 n23) 7) (=
(travel-slow n21 n24) 8) (= (travel-slow n21 n25) 9) (=
(travel-slow n21 n26) 10) (= (travel-slow n21 n27) 11) (=
(travel-slow n21 n28) 12) (= (travel-slow n21 n29) 13) (=
(travel-slow n21 n30) 14) (= (travel-slow n22 n23) 6) (=
(travel-slow n22 n24) 7) (= (travel-slow n22 n25) 8) (=
(travel-slow n22 n26) 9) (= (travel-slow n22 n27) 10) (=
(travel-slow n22 n28) 11) (= (travel-slow n22 n29) 12) (=
(travel-slow n22 n30) 13) (= (travel-slow n23 n24) 6) (=
(travel-slow n23 n25) 7) (= (travel-slow n23 n26) 8) (=
(travel-slow n23 n27) 9) (= (travel-slow n23 n28) 10) (=
(travel-slow n23 n29) 11) (= (travel-slow n23 n30) 12) (=
(travel-slow n24 n25) 6) (= (travel-slow n24 n26) 7) (=
(travel-slow n24 n27) 8) (= (travel-slow n24 n28) 9) (=
(travel-slow n24 n29) 10) (= (travel-slow n24 n30) 11) (=
(travel-slow n25 n26) 6) (= (travel-slow n25 n27) 7) (=
(travel-slow n25 n28) 8) (= (travel-slow n25 n29) 9) (=
(travel-slow n25 n30) 10) (= (travel-slow n26 n27) 6) (=
(travel-slow n26 n28) 7) (= (travel-slow n26 n29) 8) (=
(travel-slow n26 n30) 9) (= (travel-slow n27 n28) 6) (=
(travel-slow n27 n29) 7) (= (travel-slow n27 n30) 8) (=
(travel-slow n28 n29) 6) (= (travel-slow n28 n30) 7) (=
(travel-slow n29 n30) 6)

(= (travel-slow n30 n31) 6) (= (travel-slow n30 n32) 7) (=
(travel-slow n30 n33) 8) (= (travel-slow n30 n34) 9) (=
(travel-slow n30 n35) 10) (= (travel-slow n30 n36) 11) (=
```

```
(travel-slow n30 n37) 12) (= (travel-slow n30 n38) 13) (=
(travel-slow n30 n39) 14) (= (travel-slow n30 n40) 15) (=
(travel-slow n31 n32) 6) (= (travel-slow n31 n33) 7) (=
(travel-slow n31 n34) 8) (= (travel-slow n31 n35) 9) (=
(travel-slow n31 n36) 10) (= (travel-slow n31 n37) 11) (=
(travel-slow n31 n38) 12) (= (travel-slow n31 n39) 13) (=
(travel-slow n31 n40) 14) (= (travel-slow n32 n33) 6) (=
(travel-slow n32 n34) 7) (= (travel-slow n32 n35) 8) (=
(travel-slow n32 n36) 9) (= (travel-slow n32 n37) 10) (=
(travel-slow n32 n38) 11) (= (travel-slow n32 n39) 12) (=
(travel-slow n32 n40) 13) (= (travel-slow n33 n34) 6) (=
(travel-slow n33 n35) 7) (= (travel-slow n33 n36) 8) (=
(travel-slow n33 n37) 9) (= (travel-slow n33 n38) 10) (=
(travel-slow n33 n39) 11) (= (travel-slow n33 n40) 12) (=
(travel-slow n34 n35) 6) (= (travel-slow n34 n36) 7) (=
(travel-slow n34 n37) 8) (= (travel-slow n34 n38) 9) (=
(travel-slow n34 n39) 10) (= (travel-slow n34 n40) 11) (=
(travel-slow n35 n36) 6) (= (travel-slow n35 n37) 7) (=
(travel-slow n35 n38) 8) (= (travel-slow n35 n39) 9) (=
(travel-slow n35 n40) 10) (= (travel-slow n36 n37) 6) (=
(travel-slow n36 n38) 7) (= (travel-slow n36 n39) 8) (=
(travel-slow n36 n40) 9) (= (travel-slow n37 n38) 6) (=

(travel-slow n37 n39) 7) (= (travel-slow n37 n40) 8) (=
(travel-slow n38 n39) 6) (= (travel-slow n38 n40) 7) (=
(travel-slow n39 n40) 6)


(= (travel-fast n0 n5) 16) (= (travel-fast n0 n10) 31) (=
(travel-fast n0 n15) 46) (= (travel-fast n0 n20) 61) (=
(travel-fast n0 n25) 76) (= (travel-fast n0 n30) 91) (=
(travel-fast n0 n35) 106) (= (travel-fast n0 n40) 121)

(= (travel-fast n5 n10) 16) (= (travel-fast n5 n15) 31) (=
(travel-fast n5 n20) 46) (= (travel-fast n5 n25) 61) (=
(travel-fast n5 n30) 76) (= (travel-fast n5 n35) 91) (=
(travel-fast n5 n40) 106)

(= (travel-fast n10 n15) 16) (= (travel-fast n10 n20) 31) (=
(travel-fast n10 n25) 46) (= (travel-fast n10 n30) 61) (=
(travel-fast n10 n35) 76) (= (travel-fast n10 n40) 91)

(= (travel-fast n15 n20) 16) (= (travel-fast n15 n25) 31) (=
(travel-fast n15 n30) 46) (= (travel-fast n15 n35) 61) (=
(travel-fast n15 n40) 76)

(= (travel-fast n20 n25) 16) (= (travel-fast n20 n30) 31) (=
(travel-fast n20 n35) 46) (= (travel-fast n20 n40) 61)
```

```
(= (travel-fast n25 n30) 16) (= (travel-fast n25 n35) 31) (=
(travel-fast n25 n40) 46)

(= (travel-fast n30 n35) 16) (= (travel-fast n30 n40) 31)

(= (travel-fast n35 n40) 16)

(= (total-cost) 0)

)

(:goal
(and
(passenger-at p0 n1)
(passenger-at p1 n13)
(passenger-at p2 n9)
(passenger-at p3 n18)
(passenger-at p4 n25)
(passenger-at p5 n1)
(passenger-at p6 n6)
(passenger-at p7 n31)
(passenger-at p8 n40)
(passenger-at p9 n9)
(passenger-at p10 n0)
(passenger-at p11 n30)
(passenger-at p12 n39)
(passenger-at p13 n10)
(passenger-at p14 n10)
(passenger-at p15 n14)
(passenger-at p16 n32)
(passenger-at p17 n34)
(passenger-at p18 n5)
(passenger-at p19 n13)
(passenger-at p20 n20)
(passenger-at p21 n15)
(passenger-at p22 n1)
(passenger-at p23 n16)
(passenger-at p24 n20)
(passenger-at p25 n21)
(passenger-at p26 n40)
(passenger-at p27 n37)
(passenger-at p28 n34)
(passenger-at p29 n35)
(passenger-at p30 n24)
(passenger-at p31 n24)
(passenger-at p32 n7)
(passenger-at p33 n25)
(passenger-at p34 n37)
(passenger-at p35 n21)
(passenger-at p36 n8)
```

```
(passenger-at p37 n35)
(passenger-at p38 n7)
(passenger-at p39 n31)
(passenger-at p40 n21)
(passenger-at p41 n26)
(passenger-at p42 n36)
(passenger-at p43 n4)
(passenger-at p44 n33)
(passenger-at p45 n11)
(passenger-at p46 n36)
(passenger-at p47 n32)
(passenger-at p48 n14)
(passenger-at p49 n14)
(passenger-at p50 n14)
(passenger-at p51 n28)
(passenger-at p52 n31)
(passenger-at p53 n33)
(passenger-at p54 n33)
(passenger-at p55 n25)
(passenger-at p56 n36)
(passenger-at p57 n34)
(passenger-at p58 n33)
(passenger-at p59 n21)
))

(:metric minimize (total-cost))

)
```

8.5 Elevators con modificaciones: Modif.3

8.5.1. Dominio

```
(define (domain elevators-sequencedstrips)
  (:requirements :typing :action-costs)
  (:types      elevator - object
               slow-elevator fast-elevator - elevator
               passenger - object
               count - object
         )

(:predicates
     (passenger-at ?person - passenger ?floor - count)
     (boarded ?person - passenger ?lift - elevator)
     (lift-at ?lift - elevator ?floor - count)
     (reachable-floor ?lift - elevator ?floor - count)
     (passengers ?lift - elevator ?n - count)
     (can-hold ?lift - elevator ?n - count)
     (next ?n1 - count ?n2 - count)
)

(:functions (total-cost) - number
            (travel-slow ?f1 - count ?f2 - count) - number
            (travel-fast ?f1 - count ?f2 - count) - number
            (floor-number ?floor - count) - number
)

(:action move-up-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f1)(floor-number ?f2)) (reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f1 ?f2))))

(:action move-down-slow
  :parameters (?lift - slow-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f2)(floor-number ?f1)) (reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-slow ?f2 ?f1))))

(:action move-up-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f1)(floor-number ?f2)) (reachable-floor ?lift ?f2) )
```

```
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f1 ?f2))))

(:action move-down-fast
  :parameters (?lift - fast-elevator ?f1 - count ?f2 - count
)
  :precondition (and (lift-at ?lift ?f1) (< (floor-number
?f2)(floor-number ?f1)) (reachable-floor ?lift ?f2) )
  :effect (and (lift-at ?lift ?f2) (not (lift-at ?lift ?f1))
(increase (total-cost) (travel-fast ?f2 ?f1))))

(:action board
  :parameters (?p - passenger ?lift - elevator ?f - count
?n1 - count ?n2 - count)
  :precondition (and  (lift-at ?lift ?f) (passenger-at ?p
?f) (passengers ?lift ?n1) (next ?n1 ?n2) (can-hold ?lift
?n2) )
  :effect (and (not (passenger-at ?p ?f)) (boarded ?p ?lift)
(not (passengers ?lift ?n1)) (passengers ?lift ?n2) ))

(:action leave
  :parameters (?p - passenger ?lift - elevator ?f - count
?n1 - count ?n2 - count)
  :precondition (and  (lift-at ?lift ?f) (boarded ?p ?lift)
(passengers ?lift ?n1) (next ?n2 ?n1) )
  :effect (and (passenger-at ?p ?f) (not (boarded ?p ?lift))
(not (passengers ?lift ?n1)) (passengers ?lift ?n2) ))

)
```

8.5.2. Problema 1

```
(define (problem elevators-sequencedstrips-p16_14_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16  -
count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13  - passenger
fast0 fast1  - fast-elevator
slow0-0 slow1-0 - slow-elevator
)

(:init
(next n0 n1) (next n1 n2) (next n2 n3) (next n3 n4) (next n4
n5) (next n5 n6) (next n6 n7) (next n7 n8) (next n8 n9)
(next n9 n10) (next n10 n11) (next n11 n12) (next n12 n13)
(next n13 n14) (next n14 n15) (next n15 n16)

(= (floor-number n0) 0)
(= (floor-number n1) 1)
(= (floor-number n2) 2)
(= (floor-number n3) 3)
(= (floor-number n4) 4)
(= (floor-number n5) 5)
(= (floor-number n6) 6)
(= (floor-number n7) 7)
(= (floor-number n8) 8)
(= (floor-number n9) 9)
(= (floor-number n10) 10)
(= (floor-number n11) 11)
(= (floor-number n12) 12)
(= (floor-number n13) 13)
(= (floor-number n14) 14)
(= (floor-number n15) 15)
(= (floor-number n16) 16)

(lift-at fast0 n8)
(passengers fast0 n0)
(can-hold fast0 n1) (can-hold fast0 n2) (can-hold fast0 n3)
(can-hold fast0 n4)
(reachable-floor fast0 n0)(reachable-floor fast0
n4)(reachable-floor fast0 n8)(reachable-floor fast0
n12)(reachable-floor fast0 n16)

(lift-at fast1 n12)
(passengers fast1 n0)
(can-hold fast1 n1) (can-hold fast1 n2) (can-hold fast1 n3)
(can-hold fast1 n4)
```

```
(reachable-floor fast1 n0)(reachable-floor fast1
n4)(reachable-floor fast1 n8)(reachable-floor fast1
n12)(reachable-floor fast1 n16)

(lift-at slow0-0 n2)
(passengers slow0-0 n0)
(can-hold slow0-0 n1) (can-hold slow0-0 n2) (can-hold slow0-
0 n3)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0
n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0
n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0
n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0
n7)(reachable-floor slow0-0 n8)

(lift-at slow1-0 n12)
(passengers slow1-0 n0)
(can-hold slow1-0 n1) (can-hold slow1-0 n2) (can-hold slow1-
0 n3)
(reachable-floor slow1-0 n8)(reachable-floor slow1-0
n9)(reachable-floor slow1-0 n10)(reachable-floor slow1-0
n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0
n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0
n15)(reachable-floor slow1-0 n16)

(passenger-at p0 n13)
(passenger-at p1 n10)
(passenger-at p2 n13)
(passenger-at p3 n0)
(passenger-at p4 n9)
(passenger-at p5 n12)
(passenger-at p6 n8)
(passenger-at p7 n3)
(passenger-at p8 n5)
(passenger-at p9 n2)
(passenger-at p10 n4)
(passenger-at p11 n11)
(passenger-at p12 n13)
(passenger-at p13 n7)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (=
(travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-
slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow
n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n1 n2)
6) (= (travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (=
(travel-slow n1 n5) 9) (= (travel-slow n1 n6) 10) (=
(travel-slow n1 n7) 11) (= (travel-slow n1 n8) 12) (=
(travel-slow n2 n3) 6) (= (travel-slow n2 n4) 7) (= (travel-
slow n2 n5) 8) (= (travel-slow n2 n6) 9) (= (travel-slow n2
n7) 10) (= (travel-slow n2 n8) 11) (= (travel-slow n3 n4) 6)
(= (travel-slow n3 n5) 7) (= (travel-slow n3 n6) 8) (=
```

```
(travel-slow n3 n7) 9) (= (travel-slow n3 n8) 10) (=
(travel-slow n4 n5) 6) (= (travel-slow n4 n6) 7) (= (travel-
slow n4 n7) 8) (= (travel-slow n4 n8) 9) (= (travel-slow n5
n6) 6) (= (travel-slow n5 n7) 7) (= (travel-slow n5 n8) 8)
(= (travel-slow n6 n7) 6) (= (travel-slow n6 n8) 7) (=
(travel-slow n7 n8) 6)

(= (travel-slow n8 n9) 6) (= (travel-slow n8 n10) 7) (=
(travel-slow n8 n11) 8) (= (travel-slow n8 n12) 9) (=
(travel-slow n8 n13) 10) (= (travel-slow n8 n14) 11) (=
(travel-slow n8 n15) 12) (= (travel-slow n8 n16) 13) (=
(travel-slow n9 n10) 6) (= (travel-slow n9 n11) 7) (=
(travel-slow n9 n12) 8) (= (travel-slow n9 n13) 9) (=
(travel-slow n9 n14) 10) (= (travel-slow n9 n15) 11) (=
(travel-slow n9 n16) 12) (= (travel-slow n10 n11) 6) (=
(travel-slow n10 n12) 7) (= (travel-slow n10 n13) 8) (=
(travel-slow n10 n14) 9) (= (travel-slow n10 n15) 10) (=
(travel-slow n10 n16) 11) (= (travel-slow n11 n12) 6) (=
(travel-slow n11 n13) 7) (= (travel-slow n11 n14) 8) (=
(travel-slow n11 n15) 9) (= (travel-slow n11 n16) 10) (=
(travel-slow n12 n13) 6) (= (travel-slow n12 n14) 7) (=
(travel-slow n12 n15) 8) (= (travel-slow n12 n16) 9) (=
(travel-slow n13 n14) 6) (= (travel-slow n13 n15) 7) (=
(travel-slow n13 n16) 8) (= (travel-slow n14 n15) 6) (=
(travel-slow n14 n16) 7) (= (travel-slow n15 n16) 6)


(= (travel-fast n0 n4) 13) (= (travel-fast n0 n8) 25) (=
(travel-fast n0 n12) 37) (= (travel-fast n0 n16) 49)

(= (travel-fast n4 n8) 13) (= (travel-fast n4 n12) 25) (=
(travel-fast n4 n16) 37)

(= (travel-fast n8 n12) 13) (= (travel-fast n8 n16) 25)

(= (travel-fast n12 n16) 13)

(= (total-cost) 0)

)

(:goal
(and
(passenger-at p0 n8)
(passenger-at p1 n15)
(passenger-at p2 n6)
(passenger-at p3 n14)
(passenger-at p4 n5)
(passenger-at p5 n2)
(passenger-at p6 n14)
```

```
(passenger-at p7 n4)
(passenger-at p8 n10)
(passenger-at p9 n9)
(passenger-at p10 n12)
(passenger-at p11 n13)
(passenger-at p12 n5)
(passenger-at p13 n6)
))

(:metric minimize (total-cost))

)
```

8.5.3. Problema 20

(define (problem elevators-sequencedstrips-p40_60_1)
(:domain elevators-sequencedstrips)

(:objects
n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16 n17 n18 n19 n20 n21 n22 n23 n24 n25 n26 n27 n28 n29 n30 n31 n32 n33 n34 n35 n36 n37 n38 n39 n40 - count
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16 p17 p18 p19 p20 p21 p22 p23 p24 p25 p26 p27 p28 p29 p30 p31 p32 p33 p34 p35 p36 p37 p38 p39 p40 p41 p42 p43 p44 p45 p46 p47 p48 p49 p50 p51 p52 p53 p54 p55 p56 p57 p58 p59 - passenger
fast0 fast1 fast2 fast3 - fast-elevator
slow0-0 slow1-0 slow2-0 slow3-0 - slow-elevator
)

(:init
(next n0 n1) (next n1 n2) (next n2 n3) (next n3 n4) (next n4 n5) (next n5 n6) (next n6 n7) (next n7 n8) (next n8 n9) (next n9 n10) (next n10 n11) (next n11 n12) (next n12 n13) (next n13 n14) (next n14 n15) (next n15 n16) (next n16 n17) (next n17 n18) (next n18 n19) (next n19 n20) (next n20 n21) (next n21 n22) (next n22 n23) (next n23 n24) (next n24 n25) (next n25 n26) (next n26 n27) (next n27 n28) (next n28 n29) (next n29 n30) (next n30 n31) (next n31 n32) (next n32 n33) (next n33 n34) (next n34 n35) (next n35 n36) (next n36 n37) (next n37 n38) (next n38 n39) (next n39 n40)

(= (floor-number n0) 0)
(= (floor-number n1) 1)
(= (floor-number n2) 2)
(= (floor-number n3) 3)
(= (floor-number n4) 4)
(= (floor-number n5) 5)
(= (floor-number n6) 6)
(= (floor-number n7) 7)
(= (floor-number n8) 8)
(= (floor-number n9) 9)
(= (floor-number n10) 10)
(= (floor-number n11) 11)
(= (floor-number n12) 12)
(= (floor-number n13) 13)
(= (floor-number n14) 14)
(= (floor-number n15) 15)
(= (floor-number n16) 16)
(= (floor-number n17) 17)
(= (floor-number n18) 18)
(= (floor-number n19) 19)
(= (floor-number n20) 20)
(= (floor-number n21) 21)

(= (floor-number n22) 22)
(= (floor-number n23) 23)
(= (floor-number n24) 24)
(= (floor-number n25) 25)
(= (floor-number n26) 26)
(= (floor-number n27) 27)
(= (floor-number n28) 28)
(= (floor-number n29) 29)
(= (floor-number n30) 30)
(= (floor-number n31) 31)
(= (floor-number n32) 32)
(= (floor-number n33) 33)
(= (floor-number n34) 34)
(= (floor-number n35) 35)
(= (floor-number n36) 36)
(= (floor-number n37) 37)
(= (floor-number n38) 38)
(= (floor-number n39) 39)
(= (floor-number n40) 40)

(lift-at fast0 n10)
(passengers fast0 n0)
(can-hold fast0 n1) (can-hold fast0 n2) (can-hold fast0 n3) (can-hold fast0 n4) (can-hold fast0 n5) (can-hold fast0 n6)
(reachable-floor fast0 n0)(reachable-floor fast0 n5)(reachable-floor fast0 n10)(reachable-floor fast0 n15)(reachable-floor fast0 n20)(reachable-floor fast0 n25)(reachable-floor fast0 n30)(reachable-floor fast0 n35)(reachable-floor fast0 n40)

(lift-at fast1 n40)
(passengers fast1 n0)
(can-hold fast1 n1) (can-hold fast1 n2) (can-hold fast1 n3) (can-hold fast1 n4) (can-hold fast1 n5) (can-hold fast1 n6)
(reachable-floor fast1 n0)(reachable-floor fast1 n5)(reachable-floor fast1 n10)(reachable-floor fast1 n15)(reachable-floor fast1 n20)(reachable-floor fast1 n25)(reachable-floor fast1 n30)(reachable-floor fast1 n35)(reachable-floor fast1 n40)

(lift-at fast2 n0)
(passengers fast2 n0)
(can-hold fast2 n1) (can-hold fast2 n2) (can-hold fast2 n3) (can-hold fast2 n4) (can-hold fast2 n5) (can-hold fast2 n6)
(reachable-floor fast2 n0)(reachable-floor fast2 n5)(reachable-floor fast2 n10)(reachable-floor fast2 n15)(reachable-floor fast2 n20)(reachable-floor fast2 n25)(reachable-floor fast2 n30)(reachable-floor fast2 n35)(reachable-floor fast2 n40)

(lift-at fast3 n20)
(passengers fast3 n0)

(can-hold fast3 n1) (can-hold fast3 n2) (can-hold fast3 n3) (can-hold fast3 n4) (can-hold fast3 n5) (can-hold fast3 n6)
(reachable-floor fast3 n0)(reachable-floor fast3 n5)(reachable-floor fast3 n10)(reachable-floor fast3 n15)(reachable-floor fast3 n20)(reachable-floor fast3 n25)(reachable-floor fast3 n30)(reachable-floor fast3 n35)(reachable-floor fast3 n40)

(lift-at slow0-0 n9)
(passengers slow0-0 n0)
(can-hold slow0-0 n1) (can-hold slow0-0 n2) (can-hold slow0-0 n3) (can-hold slow0-0 n4)
(reachable-floor slow0-0 n0)(reachable-floor slow0-0 n1)(reachable-floor slow0-0 n2)(reachable-floor slow0-0 n3)(reachable-floor slow0-0 n4)(reachable-floor slow0-0 n5)(reachable-floor slow0-0 n6)(reachable-floor slow0-0 n7)(reachable-floor slow0-0 n8)(reachable-floor slow0-0 n9)(reachable-floor slow0-0 n10)

(lift-at slow1-0 n12)
(passengers slow1-0 n0)
(can-hold slow1-0 n1) (can-hold slow1-0 n2) (can-hold slow1-0 n3) (can-hold slow1-0 n4)
(reachable-floor slow1-0 n10)(reachable-floor slow1-0 n11)(reachable-floor slow1-0 n12)(reachable-floor slow1-0 n13)(reachable-floor slow1-0 n14)(reachable-floor slow1-0 n15)(reachable-floor slow1-0 n16)(reachable-floor slow1-0 n17)(reachable-floor slow1-0 n18)(reachable-floor slow1-0 n19)(reachable-floor slow1-0 n20)

(lift-at slow2-0 n23)
(passengers slow2-0 n0)
(can-hold slow2-0 n1) (can-hold slow2-0 n2) (can-hold slow2-0 n3) (can-hold slow2-0 n4)
(reachable-floor slow2-0 n20)(reachable-floor slow2-0 n21)(reachable-floor slow2-0 n22)(reachable-floor slow2-0 n23)(reachable-floor slow2-0 n24)(reachable-floor slow2-0 n25)(reachable-floor slow2-0 n26)(reachable-floor slow2-0 n27)(reachable-floor slow2-0 n28)(reachable-floor slow2-0 n29)(reachable-floor slow2-0 n30)

(lift-at slow3-0 n36)
(passengers slow3-0 n0)
(can-hold slow3-0 n1) (can-hold slow3-0 n2) (can-hold slow3-0 n3) (can-hold slow3-0 n4)
(reachable-floor slow3-0 n30)(reachable-floor slow3-0 n31)(reachable-floor slow3-0 n32)(reachable-floor slow3-0 n33)(reachable-floor slow3-0 n34)(reachable-floor slow3-0 n35)(reachable-floor slow3-0 n36)(reachable-floor slow3-0 n37)(reachable-floor slow3-0 n38)(reachable-floor slow3-0 n39)(reachable-floor slow3-0 n40)

(passenger-at p0 n18)
(passenger-at p1 n32)
(passenger-at p2 n17)
(passenger-at p3 n32)
(passenger-at p4 n35)
(passenger-at p5 n3)
(passenger-at p6 n38)
(passenger-at p7 n38)

(passenger-at p8 n36)
(passenger-at p9 n19)
(passenger-at p10 n34)
(passenger-at p11 n19)
(passenger-at p12 n2)
(passenger-at p13 n25)
(passenger-at p14 n13)
(passenger-at p15 n29)
(passenger-at p16 n1)
(passenger-at p17 n11)
(passenger-at p18 n9)
(passenger-at p19 n19)
(passenger-at p20 n8)
(passenger-at p21 n11)
(passenger-at p22 n17)
(passenger-at p23 n11)
(passenger-at p24 n21)
(passenger-at p25 n10)
(passenger-at p26 n39)
(passenger-at p27 n24)
(passenger-at p28 n25)
(passenger-at p29 n9)
(passenger-at p30 n22)
(passenger-at p31 n37)
(passenger-at p32 n15)
(passenger-at p33 n18)
(passenger-at p34 n12)
(passenger-at p35 n38)
(passenger-at p36 n17)
(passenger-at p37 n36)
(passenger-at p38 n10)
(passenger-at p39 n10)
(passenger-at p40 n27)
(passenger-at p41 n12)
(passenger-at p42 n20)
(passenger-at p43 n23)
(passenger-at p44 n29)
(passenger-at p45 n40)
(passenger-at p46 n16)
(passenger-at p47 n36)
(passenger-at p48 n2)
(passenger-at p49 n16)
(passenger-at p50 n11)
(passenger-at p51 n35)
(passenger-at p52 n23)
(passenger-at p53 n22)

(passenger-at p54 n38)
(passenger-at p55 n24)
(passenger-at p56 n13)
(passenger-at p57 n10)
(passenger-at p58 n32)
(passenger-at p59 n38)

(= (travel-slow n0 n1) 6) (= (travel-slow n0 n2) 7) (= (travel-slow n0 n3) 8) (= (travel-slow n0 n4) 9) (= (travel-slow n0 n5) 10) (= (travel-slow n0 n6) 11) (= (travel-slow n0 n7) 12) (= (travel-slow n0 n8) 13) (= (travel-slow n0 n9) 14) (= (travel-slow n0 n10) 15) (= (travel-slow n1 n2) 6) (= (travel-slow n1 n3) 7) (= (travel-slow n1 n4) 8) (= (travel-slow n1 n5) 9) (= (travel-slow n1 n6) 10) (= (travel-slow n1 n7) 11) (= (travel-slow n1 n8) 12) (= (travel-slow n1 n9) 13) (= (travel-slow n1 n10) 14) (= (travel-slow n2 n3) 6) (= (travel-slow n2 n4) 7) (= (travel-slow n2 n5) 8) (= (travel-slow n2 n6) 9) (= (travel-slow n2 n7) 10) (= (travel-slow n2 n8) 11) (= (travel-slow n2 n9) 12) (= (travel-slow n2 n10) 13) (= (travel-slow n3 n4) 6) (= (travel-slow n3 n5) 7) (= (travel-slow n3 n6) 8) (= (travel-slow n3 n7) 9) (= (travel-slow n3 n8) 10) (= (travel-slow n3 n9) 11) (= (travel-slow n3 n10) 12) (= (travel-slow n4 n5) 6) (= (travel-slow n4 n6) 7) (= (travel-slow n4 n7) 8) (= (travel-slow n4 n8) 9) (= (travel-slow n4 n9) 10) (= (travel-slow n4 n10) 11) (= (travel-slow n5 n6) 6) (= (travel-slow n5 n7) 7) (= (travel-slow n5 n8) 8) (= (travel-slow n5 n9) 9) (= (travel-slow n5 n10) 10) (= (travel-slow n6 n7) 6) (= (travel-slow n6 n8) 7) (= (travel-slow n6 n9) 8) (= (travel-slow n6 n10) 9) (= (travel-slow n7 n8) 6) (= (travel-slow n7 n9) 7) (= (travel-slow n7 n10) 8) (= (travel-slow n8 n9) 6) (= (travel-slow n8 n10) 7) (= (travel-slow n9 n10) 6)

(= (travel-slow n10 n11) 6) (= (travel-slow n10 n12) 7) (= (travel-slow n10 n13) 8) (= (travel-slow n10 n14) 9) (= (travel-slow n10 n15) 10) (= (travel-slow n10 n16) 11) (= (travel-slow n10 n17) 12) (= (travel-slow n10 n18) 13) (= (travel-slow n10 n19) 14) (= (travel-slow n10 n20) 15) (= (travel-slow n11 n12) 6) (= (travel-slow n11 n13) 7) (= (travel-slow n11 n14) 8) (= (travel-slow n11 n15) 9) (= (travel-slow n11 n16) 10) (= (travel-slow n11 n17) 11) (= (travel-slow n11 n18) 12) (= (travel-slow n11 n19) 13) (= (travel-slow n11 n20) 14) (= (travel-slow n12 n13) 6) (= (travel-slow n12 n14) 7) (= (travel-slow n12 n15) 8) (= (travel-slow n12 n16) 9) (= (travel-slow n12 n17) 10) (= (travel-slow n12 n18) 11) (= (travel-slow n12 n19) 12) (= (travel-slow n12 n20) 13) (= (travel-slow n13 n14) 6) (= (travel-slow n13 n15) 7) (= (travel-slow n13 n16) 8) (= (travel-slow n13 n17) 9) (= (travel-slow n13 n18) 10) (= (travel-slow n13 n19) 11) (= (travel-slow n13 n20) 12) (= (travel-slow n14 n15) 6) (= (travel-slow n14 n16) 7) (= (travel-slow n14 n17) 8) (= (travel-slow n14 n18) 9) (= (travel-slow n14 n19) 10) (= (travel-slow n14 n20) 11) (= (travel-slow n15 n16) 6) (= (travel-slow n15 n17) 7) (= (travel-slow n15 n18) 8) (= (travel-slow n15 n19) 9) (= (travel-slow n15 n20) 10) (= (travel-slow n16 n17) 6) (= (travel-slow n16 n18) 7) (= (travel-slow n16 n19) 8) (= (travel-slow n16 n20) 9) (= (travel-slow n17 n18) 6) (= (travel-slow n17 n19) 7) (= (travel-slow n17 n20) 8) (= (travel-slow n18 n19) 6) (= (travel-slow n18 n20) 7) (= (travel-slow n19 n20) 6)

(= (travel-slow n20 n21) 6) (= (travel-slow n20 n22) 7) (= (travel-slow n20 n23) 8) (= (travel-slow n20 n24) 9) (= (travel-slow n20 n25) 10) (= (travel-slow n20 n26) 11) (= (travel-slow n20 n27) 12) (= (travel-slow n20 n28) 13) (= (travel-slow n20 n29) 14) (=

(travel-slow n20 n30) 15) (= (travel-slow n21 n22) 6) (= (travel-slow n21 n23) 7) (= (travel-slow n21 n24) 8) (= (travel-slow n21 n25) 9) (= (travel-slow n21 n26) 10) (= (travel-slow n21 n27) 11) (= (travel-slow n21 n28) 12) (= (travel-slow n21 n29) 13) (= (travel-slow n21 n30) 14) (= (travel-slow n22 n23) 6) (= (travel-slow n22 n24) 7) (= (travel-slow n22 n25) 8) (= (travel-slow n22 n26) 9) (= (travel-slow n22 n27) 10) (= (travel-slow n22 n28) 11) (= (travel-slow n22 n29) 12) (= (travel-slow n22 n30) 13) (= (travel-slow n23 n24) 6) (= (travel-slow n23 n25) 7) (= (travel-slow n23 n26) 8) (= (travel-slow n23 n27) 9) (= (travel-slow n23 n28) 10) (= (travel-slow n23 n29) 11) (= (travel-slow n23 n30) 12) (= (travel-slow n24 n25) 6) (= (travel-slow n24 n26) 7) (= (travel-slow n24 n27) 8) (= (travel-slow n24 n28) 9) (= (travel-slow n24 n29) 10) (= (travel-slow n24 n30) 11) (= (travel-slow n25 n26) 6) (= (travel-slow n25 n27) 7) (= (travel-slow n25 n28) 8) (= (travel-slow n25 n29) 9) (= (travel-slow n25 n30) 10) (= (travel-slow n26 n27) 6) (= (travel-slow n26 n28) 7) (= (travel-slow n26 n29) 8) (= (travel-slow n26 n30) 9) (= (travel-slow n27 n28) 6) (= (travel-slow n27 n29) 7) (= (travel-slow n27 n30) 8) (= (travel-slow n28 n29) 6) (= (travel-slow n28 n30) 7) (= (travel-slow n29 n30) 6)

(= (travel-slow n30 n31) 6) (= (travel-slow n30 n32) 7) (= (travel-slow n30 n33) 8) (= (travel-slow n30 n34) 9) (= (travel-slow n30 n35) 10) (= (travel-slow n30 n36) 11) (= (travel-slow n30 n37) 12) (= (travel-slow n30 n38) 13) (= (travel-slow n30 n39) 14) (= (travel-slow n30 n40) 15) (= (travel-slow n31 n32) 6) (= (travel-slow n31 n33) 7) (= (travel-slow n31 n34) 8) (= (travel-slow n31 n35) 9) (= (travel-slow n31 n36) 10) (= (travel-slow n31 n37) 11) (= (travel-slow n31 n38) 12) (= (travel-slow n31 n39) 13) (= (travel-slow n31 n40) 14) (= (travel-slow n32 n33) 6) (= (travel-slow n32 n34) 7) (= (travel-slow n32 n35) 8) (= (travel-slow n32 n36) 9) (= (travel-slow n32 n37) 10) (= (travel-slow n32 n38) 11) (= (travel-slow n32 n39) 12) (= (travel-slow n32 n40) 13) (= (travel-slow n33 n34) 6) (= (travel-slow n33 n35) 7) (= (travel-slow n33 n36) 8) (= (travel-slow n33 n37) 9) (= (travel-slow n33 n38) 10) (= (travel-slow n33 n39) 11) (= (travel-slow n33 n40) 12) (= (travel-slow n34 n35) 6) (= (travel-slow n34 n36) 7) (= (travel-slow n34 n37) 8) (= (travel-slow n34 n38) 9) (= (travel-slow n34 n39) 10) (= (travel-slow n34 n40) 11) (= (travel-slow n35 n36) 6) (= (travel-slow n35 n37) 7) (= (travel-slow n35 n38) 8) (= (travel-slow n35 n39) 9) (= (travel-slow n35 n40) 10) (= (travel-slow n36 n37) 6) (= (travel-slow n36 n38) 7) (= (travel-slow n36 n39) 8) (= (travel-slow n36 n40) 9) (= (travel-slow n37 n38) 6) (= (travel-slow n37 n39) 7) (= (travel-slow n37 n40) 8) (= (travel-slow n38 n39) 6) (= (travel-slow n38 n40) 7) (= (travel-slow n39 n40) 6)

(= (travel-fast n0 n5) 16) (= (travel-fast n0 n10) 31) (= (travel-fast n0 n15) 46) (= (travel-fast n0 n20) 61) (= (travel-fast n0 n25) 76) (= (travel-fast n0 n30) 91) (= (travel-fast n0 n35) 106) (= (travel-fast n0 n40) 121)

(= (travel-fast n5 n10) 16) (= (travel-fast n5 n15) 31) (= (travel-fast n5 n20) 46) (= (travel-fast n5 n25) 61) (= (travel-fast n5 n30) 76) (= (travel-fast n5 n35) 91) (= (travel-fast n5 n40) 106)

(= (travel-fast n10 n15) 16) (= (travel-fast n10 n20) 31) (= (travel-fast n10 n25) 46) (= (travel-fast n10 n30) 61) (= (travel-fast n10 n35) 76) (= (travel-fast n10 n40) 91)

(= (travel-fast n15 n20) 16) (= (travel-fast n15 n25) 31) (= (travel-fast n15 n30) 46) (= (travel-fast n15 n35) 61) (= (travel-fast n15 n40) 76)

(= (travel-fast n20 n25) 16) (= (travel-fast n20 n30) 31) (= (travel-fast n20 n35) 46) (= (travel-fast n20 n40) 61)

(= (travel-fast n25 n30) 16) (= (travel-fast n25 n35) 31) (= (travel-fast n25 n40) 46)

(= (travel-fast n30 n35) 16) (= (travel-fast n30 n40) 31)

(= (travel-fast n35 n40) 16)

(= (total-cost) 0)

)

(:goal
(and
(passenger-at p0 n1)
(passenger-at p1 n13)
(passenger-at p2 n9)
(passenger-at p3 n18)
(passenger-at p4 n25)
(passenger-at p5 n1)
(passenger-at p6 n6)
(passenger-at p7 n31)
(passenger-at p8 n40)
(passenger-at p9 n9)
(passenger-at p10 n0)
(passenger-at p11 n30)
(passenger-at p12 n39)
(passenger-at p13 n10)
(passenger-at p14 n10)
(passenger-at p15 n14)
(passenger-at p16 n32)
(passenger-at p17 n34)
(passenger-at p18 n5)
(passenger-at p19 n13)
(passenger-at p20 n20)
(passenger-at p21 n15)
(passenger-at p22 n1)
(passenger-at p23 n16)
(passenger-at p24 n20)
(passenger-at p25 n21)
(passenger-at p26 n40)
(passenger-at p27 n37)

```
(passenger-at p28 n34)
(passenger-at p29 n35)
(passenger-at p30 n24)
(passenger-at p31 n24)
(passenger-at p32 n7)
(passenger-at p33 n25)
(passenger-at p34 n37)
(passenger-at p35 n21)
(passenger-at p36 n8)
(passenger-at p37 n35)
(passenger-at p38 n7)
(passenger-at p39 n31)
(passenger-at p40 n21)
(passenger-at p41 n26)
(passenger-at p42 n36)
(passenger-at p43 n4)
(passenger-at p44 n33)
(passenger-at p45 n11)
(passenger-at p46 n36)
(passenger-at p47 n32)
(passenger-at p48 n14)
(passenger-at p49 n14)
(passenger-at p50 n14)
(passenger-at p51 n28)
(passenger-at p52 n31)
(passenger-at p53 n33)
(passenger-at p54 n33)
(passenger-at p55 n25)
(passenger-at p56 n36)
(passenger-at p57 n34)
(passenger-at p58 n33)
(passenger-at p59 n21)
))

(:metric minimize (total-cost))

)
```

Glosario

ACO	*Ant Colony Optimization*
ADL	*Action description Languaje*
APPL	*Abstract Plan Preparation Language*
BNF	*Backus-Naus Form*
DAE	*Divide-and-Evolve*
FAI	*Foundations of Artificial Intelligence*
FD	*Fast Downward*
IA	*Inteligencia Artificial*
ICAPS	*International Conference on Automated Planning and Scheduling*
IME	*Image Marking Engines*
IPC	*International Planning Competition*
MA-PDDL	*Multi Agent PDDL*
MAPL	*Multi-Agent Planning Language*
MDP	*Markov decision process*
NASA	National Aeronautics and Space Administration
NDDL	*New Domain Definition Language*
OPT	*Ontology with Polymorphic Types*
PDDL	*Planning Domain Description Language*
PDM	*Point Distribution Model*

POMDP	*Partially observable Markov decision process*
PPDDL	*Probabilistic Planning Domain Definition Language*
RDDL	*Relational Dynamic influence Diagram Language*
STRIPS	*Standford Research Institute Problem Solver*

Referencias

[1] Malik Ghallab, Dana Nau y Paolo Traverso: "Automated Planning: Theory & Practice", Elsevier Science & Technology (Ed.): 2004.

[2] Amanda Coles, Andrew Coles, Angel García Olaya, Sergio Jiménez, Carlos Linares López, Scott Sanner, Sungwook Yoon "A Survey of the Seventh International Planning Competition". AI MAGAZINE. Association for the Advancement of Artificial Intelligence. ISSN 0738-4602: 2012.

[3] PDDL. Representation in Planning: Transparencias asignatura de doctorado, Uc3m. Planning and Learning Group (PLG). Automated Planning 2014-2015.

[4] Malte Helmert. "An Introduction to PDDL". AI Planning. Trasparencias de introducción al lenguaje PDDL.

[5] Jan Hrnčíř. "A4M33PAH Tutorial". PDDL (Planning Domain Definition Language). Trasparencias de introducción al lenguaje PDDL.

[6] Carlos Linares López, Sergio Jiménez Celorrio, Ángel García Olaya. "The deterministic part of the seventh International Planning Competition". Artificial Intelligence. Elsevier B.V. (Ed.): 2015.

[7] Carlos Linares López, Sergio Jiménez Celorrio, Ángel García Olaya. "The deterministic part of the seventh International Planning Competition – Appendices". Artificial Intelligence. Elsevier B.V. (Ed.): 2015.

[8] Dana S. Nau. "Automated Planning: Theory and Practice". Lecture slides for Automated Planning. University of Maryland, 2012.

[9] Maria Fox, Derek Long. "An Extension to PPDL for Expressing Temporal Planning Domains". Journal of Artificial Intelligence Research 20, 2003.

Referencias web:

- Web del grupo de Planificación y Aprendizaje – UC3M. http://www.plg.inf.uc3m.es/
- Wikipedia.org. Planificación Automática. https://es.wikipedia.org/wiki/Planificaci%C3%B3n_autom%C3%A1tica

- Wikipedia.org. STRIPS. https://es.wikipedia.org/wiki/STRIPS
- Wikipedia.org. ADL. https://es.wikipedia.org/wiki/Action_description_language
- ICAPS: http://icaps-conference.org
- Campusvida.usc.es. Artículo "El Futuro al Microscopio…Planificación del movimiento en entornos dinámicos". Campus Vida in Blog, CITIUS. http://campusvida.usc.es/es/el-futuro-al-microscopio-planificacion-del-movimiento-en-entornos-dinamicos/
- Fast-downward.org. IPC Planners. http://www.fast-downward.org/IpcPlanners
- Web de la Saarland University. Grupo FAI. http://fai.cs.uni-saarland.de/teaching/winter13-14/planning-material/planning18-planning-systems-and-the-ipc-post-handout.pdf
- Icaps14.icaps-conference.org. ICAPS 2014. http://icaps14.icaps-conference.org/proceedings/keps/KEPS_proceedings.pdf
- Web de la Uppsala Universitet. "AI Planning-Based Service Modeling for the Internet of Things". Quentin Bahers. https://uu.diva-portal.org/smash/get/diva2:792338/FULLTEXT01.pdf
- Web TU Delft. Delft Uneversity of Technology. Planners. https://ii.tudelft.nl/trac/goal/wiki/Projects/Planning/Planners
- Web de la Aalto University - Department of Computer Science http://users.ics.aalto.fi/rintanen/jussi/satplan.html
- Web de descarga del software validador de planes automáticos: http://www.inf.kcl.ac.uk/research/groups/PLANNING/index.php?option=com_content&view=article&id=70&Itemid=77
- Web Aalto University –utilidades software de los planificadores: http://users.ics.aalto.fi/rintanen/jussi/satplan.html